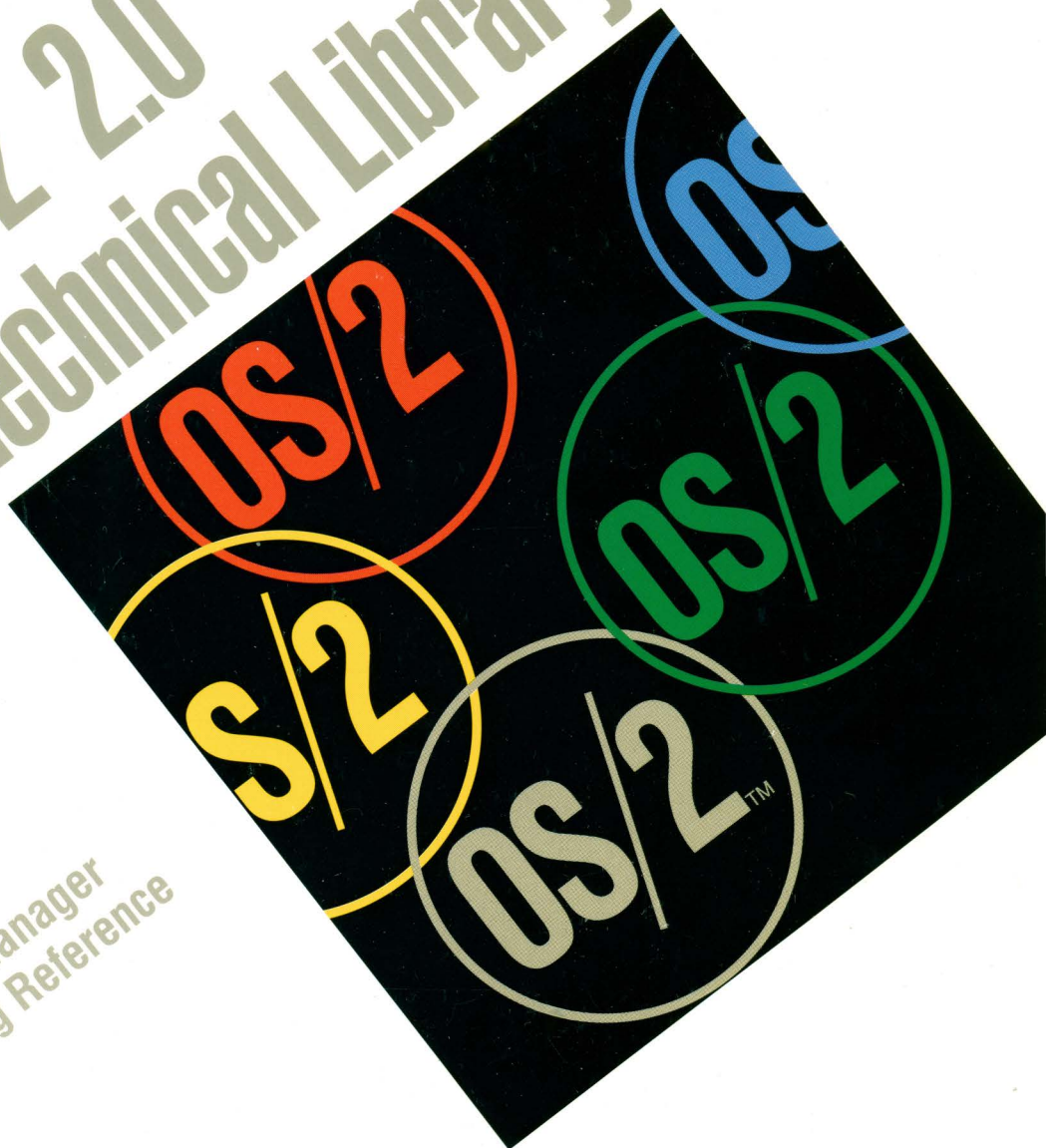


OS/2 2.0 Technical Library



Presentation Manager
Programming Reference
Volume I

Version 2.00

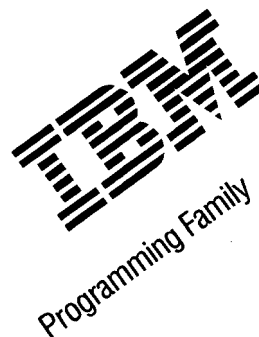


Programming Family

OS/2 2.0 Technical Library

**Presentation Manager
Programming Reference
Volume I**

Version 2.00



Note

Before using this information and the product it supports, be sure to read the general information under "Notices" on page vii.

First Edition (March 1992)

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

It is possible that this publication may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Requests for technical information about IBM products should be made to your IBM Authorized Dealer or your IBM Marketing Representative.

COPYRIGHT LICENSE: This publication contains printed sample application programs in source language, which illustrate OS/2 programming techniques. You may copy and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the OS/2 application programming interface.

Each copy of any portion of these sample programs or any derivative work, which is distributed to others, must include a copyright notice as follows: "© (your company name) (year) All Rights Reserved."

© Copyright International Business Machines Corporation 1992. All rights reserved.

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

About this Book

The *Presentation Manager Programming Reference* is a detailed technical reference, in three volumes, for application programmers creating programs using the Presentation Manager interface.

Chapter 1 contains important information. You should read it before using this book.

This reference does not give guidance on how to use the functions, nor does it contain information about how the functions are related to each other. It is intended to be used in conjunction with the *Programming Guide Volumes II and III*.

Prerequisite Knowledge

The OS/2 2.0 Technical Library is intended for professional application developers knowledgeable in at least one programming language in which OS/2 programs can be written. The information in the Technical Library assumes that you are new to programming with OS/2 and the Presentation Manager. You should understand the OS/2 services available to users.

Related Publications

The *Application Design Guide* and the *Programming Guide Volumes I, II, and III* introduce the programming concepts that you should understand before you begin developing applications to run on the OS/2 operating system. *Getting Started* describes the online programming books, tools, programming aids, and sample programs that make up the IBM Developer's Toolkit for OS/2 2.0.

Organization of this Book

This book is in three volumes. The contents of each volume are as follows:

Volume I (Functions)

Chapter 1, "Introduction" on page 1-1

You should read this chapter before using this book.

Chapter 2, "Device Functions" on page 2-1

Chapter 3, "Direct Manipulation Functions" on page 3-1

Chapter 4, "Dynamic Data Formatting Functions" on page 4-1

Chapter 5, "Graphics Functions" on page 5-1

Chapter 6, "Profile Functions" on page 6-1

Chapter 7, "Spooler Functions" on page 7-1

Volume II (Functions and Workplace)

Chapter 8, "Window Functions" on page 8-1

Chapter 9, "Workplace Classes, Instance Methods, and Class Methods" on page 9-1

Volume III (Related Information and Data Types)

Chapter 10, "Functions Supplied by Applications" on page 10-1

Chapter 11, "Introduction to Message Processing" on page 11-1

Chapter 12, "Default Window Procedure Message Processing" on page 12-1

Chapter 13, "Button Control Window Processing" on page 13-1

Chapter 14, "Entry Field Control Window Processing" on page 14-1

Chapter 15, "Frame Control Window Processing" on page 15-1

Chapter 16, "List Box Control Window Processing" on page 16-1

Chapter 17, "Menu Control Window Processing" on page 17-1

Chapter 18, "Multi-Line Entry Field Control Window Processing" on page 18-1

Chapter 19, "Prompted Entry Field Control Window Processing" on page 19-1

Chapter 20, "Scroll Bar Control Window Processing" on page 20-1

Chapter 21, "Spin Button Control Window Processing" on page 21-1

Chapter 22, "Static Control Window Processing" on page 22-1

Chapter 23, "Title Bar Control Window Processing" on page 23-1

Chapter 24, "Container Control Window Processing" on page 24-1

Chapter 25, "Notebook Control Window Processing" on page 25-1

Chapter 26, "Slider Control Window Processing" on page 26-1

Chapter 27, "Value Set Control Window Processing" on page 27-1

Chapter 28, "Clipboard Messages" on page 28-1

Chapter 29, "Direct Manipulation (Drag) Messages" on page 29-1

Chapter 30, "Dynamic Data Exchange Messages" on page 30-1

Chapter 31, "Help Manager Messages" on page 31-1

Chapter 32, "Resource Files" on page 32-1

Chapter 33, "Graphics Orders" on page 33-1

Chapter 34, “Code Pages” on page 34-1

Appendix A, “Data Types” on page A-1

Appendix B, “Error Codes” on page B-1

Appendix C, “Error Explanations” on page C-1

Appendix D, “Standard Bit-Map Formats” on page D-1

Appendix E, “Fonts Supplied with OS/2” on page E-1

Appendix F, “The Font-File Format” on page F-1

Appendix G, “Format of Interchange Files” on page G-1

Appendix H, “Initialization File Information” on page H-1

Appendix I, “Virtual Key Definitions” on page I-1

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights or other legally protectible rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by IBM, are the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

The following terms, denoted by an asterisk(*) in this publication, are trademarks of the IBM Corporation in the United States and/or other countries:

IBM
Common User Access
CUA
Operating System/2
OS/2
Presentation Manager
SAA
System Application Architecture

The following terms, denoted by a double asterisk (**) in this publication, are trademarks of other companies as follows:

Adobe	Adobe Systems Incorporated
Helvetica	Linotype AG
LaserJet	Hewlett-Packard Company
Intel	Intel Corporation
Microsoft	Microsoft Corporation
PostScript	Adobe Systems Incorporated
Times New Roman	Monotype Corporation
Windows	Microsoft Corporation

Functions

Chapter 1. Introduction	1-1
Notation Conventions	1-1
Conventions used in Function Descriptions	1-1
Error Severities	1-2
Header Files	1-3
Helper Macros	1-3
Addressing Elements in Arrays	1-5
Implicit Pointer Data Types	1-5
Storage Mapping of Data Types	1-6
Double-Byte Character Set (DBCS)	1-6
 Chapter 2. Device Functions	 2-1
DevCloseDC — Close Device Context	2-2
DevEscape — Escape	2-4
DevOpenDC — Open Device Context	2-9
DevPostDeviceModes — Post Device Modes	2-12
DevQueryCaps — Query Device Capabilities	2-15
DevQueryDeviceNames — Query Device Names	2-21
DevQueryHardcopyCaps — Query Hardcopy Caps	2-24
 Chapter 3. Direct Manipulation Functions	 3-1
DrgAcceptDroppedFiles — Direct Manipulation for Files	3-2
DrgAccessDraginfo — Access Drag Information	3-4
DrgAddStrHandle — Create String Handle	3-5
DrgAllocDraginfo — Allocate DRAGINFO Structure	3-7
DrgAllocDragtransfer — Allocate DRAGTRANSFER Structures	3-9
DrgDeleteDraginfoStrHandles — Delete DRAGINFO String Handles	3-10
DrgDeleteStrHandle — Delete String Handle	3-11
DrgDrag — Drag	3-12
DrgDragFiles — Begin Dragging Files	3-16
DrgFreeDraginfo — Free DRAGINFO Structure	3-19
DrgFreeDragtransfer — Free DRAGTRANSFER Storage	3-21
DrgGetPS — Get Drag Presentation Space	3-22
DrgPostTransferMsg — Post Drag Message	3-24
DrgPushDraginfo — Access a DRAGINFO Structure	3-26
DrgQueryDragitem — Get DRAGITEM Structure	3-28
DrgQueryDragitemCount — Get Dragged Object Count	3-30
DrgQueryDragitemPtr — Get Pointer to DRAGITEM Structure	3-31
DrgQueryNativeRMF — Get Format of a Dragged Object	3-32
DrgQueryNativeRMFLen — Get String Length for Native RMF of Dragged Object	3-34
DrgQueryStrName — Get String Contents	3-36
DrgQueryStrNameLen — Get String Length	3-38
DrgQueryTrueType — Get True Type of Dragged Object	3-40
DrgQueryTrueTypeLen — Get String Length for True Type of Dragged Object	3-42
DrgReleasePS — Release Presentation Space	3-44
DrgSendTransferMsg — Send Drag Message	3-45
DrgSetDragImage — Set Drag Image	3-48
DrgSetDragitem — Set Values in DRAGITEM	3-50
DrgSetDragPointer — Set Pointing Device Pointer	3-53
DrgVerifyNativeRMF — Verify Native Rendering Mechanism and Format	3-55
DrgVerifyRMF — Verify Given Rendering Mechanism and Format	3-57
DrgVerifyTrueType — Verify True Type of Dragged Object	3-59
DrgVerifyType — Verify Type of Dragged Object	3-61
DrgVerifyTypeSet — Verify Types	3-63
 Chapter 4. Dynamic Data Formatting Functions	 4-1
DdfBeginList — Begin Definition List	4-2
DdfBitmap — Place Bitmap Reference	4-5
DdfEndList — End Definition List	4-8

DdfHyperText	Define Hypertext Link	4-10
DdfInform	Define Inform Link	4-13
DdfInitialize	Initialize DDF Area	4-15
DdfListItem	Insert List Item	4-18
DdfMetafile	Place Metafile Reference	4-21
DdfPara	Create a Paragraph in DDF Buffer	4-24
DdfSetColor	Set Color of Text	4-26
DdfSetFont	Specify Text Font	4-29
DdfSetFontStyle	Specify Text Font Style	4-32
DdfSetFormat	Control Formatting	4-35
DdfSetTextAlign	Define Text Alignment	4-37
DdfText	Add Text to DDF Buffer	4-39

Chapter 5. Graphics Functions		5-1
Coordinates		5-1
Matrix Parameter Values		5-1
Rounding Errors		5-1
GPI Functions by Functional Area		5-2
GpiAnimatePalette	Animate Palette	5-8
GpiAssociate	Associate	5-11
GpiBeginArea	Begin Area	5-13
GpiBeginElement	Begin Element	5-17
GpiBeginPath	Begin Path	5-19
GpiBitBlt	Bit Bit	5-23
GpiBox	Box	5-28
GpiCallSegmentMatrix	Call Segment Matrix	5-31
GpiCharString	Character String	5-34
GpiCharStringAt	Character String At	5-36
GpiCharStringPos	Character String Position	5-39
GpiCharStringPosAt	Character String Position At	5-42
GpiCloseFigure	Close Figure	5-45
GpiCloseSegment	Close Segment	5-47
GpiCombineRegion	Combine Region	5-49
GpiComment	Comment	5-51
GpiConvert	Convert	5-53
GpiConvertWithMatrix	Convert with Matrix	5-55
GpiCopyMetaFile	Copy Metafile	5-57
GpiCorrelateChain	Correlate Chain	5-59
GpiCorrelateFrom	Correlate From	5-63
GpiCorrelateSegment	Correlate Segment	5-67
GpiCreateBitmap	Create Bit Map	5-71
GpiCreateLogColorTable	Create Logical Color Table	5-74
GpiCreateLogFont	Create Logical Font	5-78
GpiCreatePalette	Create Palette	5-81
GpiCreatePS	Create Presentation Space	5-84
GpiCreateRegion	Create Region	5-88
GpiDeleteBitmap	Delete Bit Map	5-90
GpiDeleteElement	Delete Element	5-92
GpiDeleteElementRange	Delete Element Range	5-94
GpiDeleteElementsBetweenLabels	Delete Elements Between Labels	5-96
GpiDeleteMetaFile	Delete Metafile	5-98
GpiDeletePalette	Delete Palette	5-100
GpiDeleteSegment	Delete Segment	5-102
GpiDeleteSegments	Delete Segments	5-104
GpiDeleteSetId	Delete Set Identifier	5-106
GpiDestroyPS	Destroy Presentation Space	5-108
GpiDestroyRegion	Destroy Region	5-110
GpiDrawBits	Draw Bits	5-112
GpiDrawChain	Draw Chain	5-117
GpiDrawDynamics	Draw Dynamics	5-119
GpiDrawFrom	Draw From	5-121
GpiDrawSegment	Draw Segment	5-123
GpiElement	Element	5-125

GpiEndArea — End Area	5-128
GpiEndElement — End Element	5-130
GpiEndPath — End Path	5-132
GpiEqualRegion — Equal Region	5-134
GpiErase — Erase	5-136
GpiErrorSegmentData — Error Segment Data	5-138
GpiExcludeClipRectangle — Exclude Clip Rectangle	5-140
GpiFillPath — Fill Path	5-142
GpiFloodFill — Flood Fill	5-144
GpiFrameRegion — Frame Region	5-146
GpiFullArc — Full Arc	5-148
GpiGetData — Get Data	5-150
GpiImage — Image	5-153
GpiIntersectClipRectangle — Intersect Clip Rectangle	5-155
GpiLabel — Label	5-157
GpiLine — Line	5-159
GpiLoadBitmap — Load Bit Map	5-161
GpiLoadFonts — Load Fonts	5-163
GpiLoadMetaFile — Load Metafile	5-165
GpiLoadPublicFonts — Load Public Fonts	5-167
GpiMarker — Marker	5-168
GpiModifyPath — Modify Path	5-170
GpiMove — Move	5-173
GpiOffsetClipRegion — Offset Clip Region	5-175
GpiOffsetElementPointer — Offset Element Pointer	5-177
GpiOffsetRegion — Offset Region	5-179
GpiOpenSegment — Open Segment	5-181
GpiOutlinePath — Outline Path	5-184
GpiPaintRegion — Paint Region	5-186
GpiPartialArc — Partial Arc	5-188
GpiPathToRegion — Path to Region	5-191
GpiPlayMetaFile — Play Metafile	5-193
GpiPointArc — Point Arc	5-199
GpiPolyFillet — Polyfillet	5-201
GpiPolyFilletSharp — Polyfillet Sharp	5-204
GpiPolygons — Draw Polygons	5-207
GpiPolyLine — Polyline	5-209
GpiPolyLineDisjoint — Polyline Disjoint	5-211
GpiPolyMarker — Polymarker	5-213
GpiPolySpline — Polyspline	5-215
GpiPop — Pop	5-217
GpiPtInRegion — Point In Region	5-219
GpiPtVisible — Point Visible	5-221
GpiPutData — Put Data	5-223
GpiQueryArcParams — Query Arc Parameters	5-226
GpiQueryAttrMode — Query Attribute Mode	5-228
GpiQueryAttrs — Query Attributes	5-229
GpiQueryBackColor — Query Background Color	5-231
GpiQueryBackMix — Query Background Mix	5-232
GpiQueryBitmapBits — Query Bit-Map Bits	5-233
GpiQueryBitmapDimension — Query Bit-Map Dimension	5-236
GpiQueryBitmapInfoHeader — Query Bit-Map Info Header	5-237
GpiQueryBitmapHandle — Query Bit-Map Handle	5-239
GpiQueryBitmapParameters — Query Bit-Map Parameters	5-240
GpiQueryBoundaryData — Query Boundary Data	5-242
GpiQueryCharAngle — Query Character Angle	5-244
GpiQueryCharBox — Query Character Box	5-246
GpiQueryCharBreakExtra — Query Character Break Extra	5-248
GpiQueryCharDirection — Query Character Direction	5-249
GpiQueryCharExtra — Query Character Extra	5-250
GpiQueryCharMode — Query Character Mode	5-251
GpiQueryCharSet — Query Character Set	5-252
GpiQueryCharShear — Query Character Shear	5-253

GpiQueryCharStringPos	— Query Character String Positions	5-255
GpiQueryCharStringPosAt	— Query Character String Positions At	5-257
GpiQueryClipBox	— Query Clip Box	5-259
GpiQueryClipRegion	— Query Clip Region	5-261
GpiQueryColor	— Query Color	5-262
GpiQueryColorData	— Query Color Data	5-264
GpiQueryColorIndex	— Query Color Index	5-266
GpiQueryCp	— Query Code Page	5-268
GpiQueryCurrentPosition	— Query Current Position	5-269
GpiQueryDefArcParams	— Query Default Arc Parameters	5-270
GpiQueryDefAttrs	— Query Default Attributes	5-271
GpiQueryDefaultViewMatrix	— Query Default View Matrix	5-273
GpiQueryDefCharBox	— Query Default Graphics Character Box	5-275
GpiQueryDefTag	— Query Default Tag	5-277
GpiQueryDefViewingLimits	— Query Default Viewing Limits	5-278
GpiQueryDevice	— Query Device	5-279
GpiQueryDeviceBitmapFormats	— Query Device Bit-Map Formats	5-280
GpiQueryDrawControl	— Query Draw Control	5-282
GpiQueryDrawingMode	— Query Drawing Mode	5-284
GpiQueryEditMode	— Query Edit Mode	5-285
GpiQueryElement	— Query Element	5-286
GpiQueryElementPointer	— Query Element Pointer	5-288
GpiQueryElementType	— Query Element Type	5-290
GpiQueryFaceString	— Query Face String	5-292
GpiQueryFontAction	— Query Font Action	5-294
GpiQueryFontFileDescriptions	— Query Font File Descriptions	5-295
GpiQueryFontMetrics	— Query Font Metrics	5-297
GpiQueryFonts	— Query Fonts	5-299
GpiQueryFullFontFileDescriptions	— Query Full Font File Descriptions	5-301
GpiQueryGraphicsField	— Query Graphics Field	5-303
GpiQueryInitialSegmentAttrs	— Query Initial Segment Attributes	5-304
GpiQueryKerningPairs	— Query Kerning Pairs	5-306
GpiQueryLineEnd	— Query Line End	5-308
GpiQueryLineJoin	— Query Line Join	5-309
GpiQueryLineType	— Query Line Type	5-310
GpiQueryLineWidth	— Query Line Width	5-311
GpiQueryLineWidthGeom	— Query Line Width Geom	5-312
GpiQueryLogColorTable	— Query Logical Color Table	5-313
GpiQueryLogicalFont	— Query Logical Font	5-315
GpiQueryMarker	— Query Marker	5-317
GpiQueryMarkerBox	— Query Marker Box	5-318
GpiQueryMarkerSet	— Query Marker Set	5-320
GpiQueryMetaFileBits	— Query Metafile Bits	5-321
GpiQueryMetaFileLength	— Query Metafile Length	5-323
GpiQueryMix	— Query Mix	5-324
GpiQueryModelTransformMatrix	— Query Model Transform Matrix	5-325
GpiQueryNearestColor	— Query Nearest Color	5-327
GpiQueryNumberSetIds	— Query Number Set Identifiers	5-329
GpiQueryPageViewport	— Query Page Viewport	5-330
GpiQueryPalette	— Query Palette	5-332
GpiQueryPaletteInfo	— Query Palette Info	5-333
GpiQueryPattern	— Query Pattern	5-335
GpiQueryPatternRefPoint	— Query Pattern Reference Point	5-336
GpiQueryPatternSet	— Query Pattern Set	5-337
GpiQueryPel	— Query Pel	5-338
GpiQueryPickAperturePosition	— Query Pick Aperture Position	5-340
GpiQueryPickApertureSize	— Query Pick Aperture Size	5-341
GpiQueryPS	— Query Presentation Space	5-342
GpiQueryRealColors	— Query Real Colors	5-343
GpiQueryRegionBox	— Query Region Box	5-345
GpiQueryRegionRects	— Query Region Rectangles	5-347
GpiQueryRGBColor	— Query RGB Color	5-349
GpiQuerySegmentAttrs	— Query Segment Attributes	5-351

GpiQuerySegmentNames	Query Segment Names	5-353
GpiQuerySegmentPriority	Query Segment Priority	5-355
GpiQuerySegmentTransformMatrix	Query Segment Transform Matrix	5-357
GpiQuerySetIds	Query Set Identifiers	5-359
GpiQueryStopDraw	Query Stop Draw	5-362
GpiQueryTag	Query Tag	5-363
GpiQueryTextAlignment	Query Text Alignment	5-364
GpiQueryTextBox	Query Text Box	5-365
GpiQueryViewingLimits	Query Viewing Limits	5-368
GpiQueryViewingTransformMatrix	Query Viewing Transform Matrix	5-370
GpiQueryWidthTable	Query Font Width Table	5-372
GpiRectInRegion	Rectangle In Region	5-374
GpiRectVisible	Rectangle Visible	5-376
GpiRemoveDynamics	Remove Dynamics	5-378
GpiResetBoundaryData	Reset Boundary Data	5-381
GpiResetPS	Reset Presentation Space	5-382
GpiRestorePS	Restore Presentation Space	5-384
GpiRotate	Rotate Transform	5-386
GpiSaveMetaFile	Save Metafile	5-389
GpiSavePS	Save Presentation Space	5-391
GpiScale	Scale Matrix	5-393
GpiSelectPalette	Select Palette	5-396
GpiSetArcParams	Set Arc Parameters	5-398
GpiSetAttrMode	Set Attribute Mode	5-401
GpiSetAttrs	Set Attributes	5-404
GpiSetBackColor	Set Background Color	5-412
GpiSetBackMix	Set Background Mix	5-415
GpiSetBitmap	Set Bit Map	5-418
GpiSetBitmapBits	Set Bit-Map Bits	5-420
GpiSetBitmapDimension	Set Bit-Map Dimension	5-423
GpiSetBitmapId	Set Bit-Map Identifier	5-425
GpiSetCharAngle	Set Character Angle	5-427
GpiSetCharBox	Set Character Box	5-430
GpiSetCharBreakExtra	Set Character Break Extra	5-433
GpiSetCharDirection	Set Character Direction	5-435
GpiSetCharExtra	Set Character Extra	5-438
GpiSetCharMode	Set Character Mode	5-440
GpiSetCharSet	Set Character Set	5-443
GpiSetCharShear	Set Character Shear	5-445
GpiSetClipPath	Set Clip Path	5-448
GpiSetClipRegion	Set Clip Region	5-451
GpiSetColor	Set Color	5-453
GpiSetCp	Set Code Page	5-456
GpiSetCurrentPosition	Set Current Position	5-458
GpiSetDefArcParams	Set Default Arc Parameters	5-460
GpiSetDefAttrs	Set Default Attributes	5-462
GpiSetDefaultViewMatrix	Set Default View Matrix	5-467
GpiSetDefTag	Set Default Tag	5-470
GpiSetDefViewingLimits	Set Default Viewing Limits	5-472
GpiSetDrawControl	Set Draw Control	5-474
GpiSetDrawingMode	Set Drawing Mode	5-477
GpiSetEditMode	Set Edit Mode	5-480
GpiSetElementPointer	Set Element Pointer	5-482
GpiSetElementPointerAtLabel	Set Element Pointer At Label	5-484
GpiSetGraphicsField	Set Graphics Field	5-486
GpiSetInitialSegmentAttrs	Set Initial Segment Attributes	5-488
GpiSetLineEnd	Set Line End	5-491
GpiSetLineJoin	Set Line Join	5-493
GpiSetLineType	Set Line Type	5-495
GpiSetLineWidth	Set Line Width	5-498
GpiSetLineWidthGeom	Set Line Width Geom	5-500
GpiSetMarker	Set Marker	5-502
GpiSetMarkerBox	Set Marker Box	5-504

GpiSetMarkerSet — Set Marker Set	5-506
GpiSetMetaFileBits — Set Metafile Bits	5-508
GpiSetMix — Set Mix	5-510
GpiSetModelTransformMatrix — Set Model Transform Matrix	5-513
GpiSetPageViewport — Set Page Viewport	5-516
GpiSetPaletteEntries — Set Palette Entries	5-518
GpiSetPattern — Set Pattern	5-521
GpiSetPatternRefPoint — Set Pattern Reference Point	5-524
GpiSetPatternSet — Set Pattern Set	5-526
GpiSetPel — Set Pel	5-528
GpiSetPickAperturePosition — Set Pick-Aperture Position	5-530
GpiSetPickApertureSize — Set Pick-Aperture Size	5-531
GpiSetPS — Set Presentation Space	5-533
GpiSetRegion — Set Region	5-536
GpiSetSegmentAttrs — Set Segment Attributes	5-538
GpiSetSegmentPriority — Set Segment Priority	5-541
GpiSetSegmentTransformMatrix — Set Segment Transform Matrix	5-543
GpiSetStopDraw — Set Stop Draw	5-546
GpiSetTag — Set Tag	5-548
GpiSetTextAlignment — Set Text Alignment	5-550
GpiSetViewingLimits — Set Viewing Limits	5-553
GpiSetViewingTransformMatrix — Set Viewing Transform Matrix	5-555
GpiStrokePath — Stroke Path	5-558
GpiTranslate — Translate Matrix	5-560
GpiUnloadFonts — Unload Fonts	5-563
GpiUnloadPublicFonts — Unload Public Fonts	5-565
GpiWCBitBit — World Coordinates Bit Bit	5-567

Chapter 6. Profile Functions	6-1
PrfCloseProfile — Close Profile	6-2
PrfOpenProfile — Open Profile	6-3
PrfQueryProfile — Query Profile	6-5
PrfQueryProfileData — Query Profile Data	6-7
PrfQueryProfileInt — Query Profile Integer	6-10
PrfQueryProfileSize — Query Profile Size	6-12
PrfQueryProfileString — Query Profile String	6-14
PrfReset — Reset Presentation Manager	6-17
PrfWriteProfileData — Write Profile Data	6-19
PrfWriteProfileString — Write Profile String	6-21

Chapter 7. Spooler Functions	7-1
SplControlDevice — Spooler Control Device	7-2
SplCopyJob — Spooler Copy Job	7-5
SplCreateDevice — Spooler Create Device	7-7
SplCreateQueue — Spooler Create Queue	7-10
SplDeleteDevice — Spooler Delete Device	7-14
SplDeleteJob — Spooler Delete Job	7-16
SplDeleteQueue — Spooler Delete Queue	7-18
SplEnumDevice — Spooler Enumerate Device	7-20
SplEnumDriver — Spooler Enumerate Driver	7-23
SplEnumJob — Spooler Enumerate Job	7-26
SplEnumPort — Spooler Enumerate Port	7-29
SplEnumPrinter — Spooler Enumerate Print Destinations	7-32
SplEnumQueue — Spooler Enumerate Queue	7-35
SplEnumQueueProcessor — Spooler Enumerate Queue Processor	7-39
SplHoldJob — Spooler Hold Job	7-42
SplHoldQueue — Spooler Hold Queue	7-44
SplPurgeQueue — Spooler Purge Queue	7-46
SplQmAbort — Spooler File Abort	7-48
SplQmAbortDoc — Spooler File Abort Document	7-49
SplQmClose — Spool File Close	7-50
SplQmEndDoc — Spooler File End Document	7-51
SplQmOpen — Spooler File Open	7-53

SplQmStartDoc	— Spooler File Start Document	7-55
SplQmWrite	— Spooler File Write	7-57
SplQueryDevice	— Spooler Query Device	7-59
SplQueryJob	— Spooler Query Job	7-62
SplQueryQueue	— Spooler Query Queue	7-66
SplReleaseJob	— Spooler Release Job	7-70
SplReleaseQueue	— Spooler Release Queue	7-72
SplSetDevice	— Spooler Set Device	7-74
SplSetJob	— Spooler Set Job	7-77
SplSetQueue	— Spooler Set Queue	7-81

Chapter 1. Introduction

This chapter contains important information. Read it before using this book.

The purpose of this reference is to give important information about functions, messages, constants, error codes, and data types. It provides language-dependent information about the functions which enables the user to generate call statements in C Language.

The following information is provided:

- The parameter list for each function.
- The syntax of each data type and structure

Notation Conventions

The following notation conventions are used in this reference:

NULL	The term NULL applied to a parameter is used to indicate the presence of the pointer parameter, but with no value.										
NULLHANDLE	The term NULLHANDLE applied to a parameter is used to indicate the presence of the handle parameter, but with no value.										
Implicit Pointer	If no entry for a data type "Pxxxxxx" is found in Appendix A, "Data Types" on page A-1, then it is implicitly a pointer to the data type "xxxxxx." See "Implicit Pointer Data Types" on page 1-5.										
Constant Names	All constants are written in uppercase. Where applicable, constant names have a prefix derived from the name of a function, message, or idea associated with the constant. For example: <table><tr><td>WM_CREATE</td><td>Window message</td></tr><tr><td>SV_CXICON</td><td>System value</td></tr><tr><td>CF_TEXT</td><td>Clipboard format.</td></tr></table> In this reference, a set of constants with the same prefix is written as in these examples: <table><tr><td>Window message</td><td>WM_*</td></tr><tr><td>System value</td><td>SV_*</td></tr></table>	WM_CREATE	Window message	SV_CXICON	System value	CF_TEXT	Clipboard format.	Window message	WM_*	System value	SV_*
WM_CREATE	Window message										
SV_CXICON	System value										
CF_TEXT	Clipboard format.										
Window message	WM_*										
System value	SV_*										

Conventions used in Function Descriptions

The documentation of each function contains these sections:

Function name	The function name, listed in alphabetic order of C (long) name together with the English name. This is at the top of each page followed by the name of the define that calls the correct header files to be included, the function prototype, and a brief description of the function.						
Parameters	Each parameter is listed with its data type and a brief description. There are four kinds of parameters: <table><tr><td>Input</td><td>Specified by the programmer.</td></tr><tr><td>Output</td><td>Returned by the Presentation Manager* (PM) interface.</td></tr><tr><td>Input/Output</td><td>Specified by the programmer and modified by PM.</td></tr></table>	Input	Specified by the programmer.	Output	Returned by the Presentation Manager* (PM) interface.	Input/Output	Specified by the programmer and modified by PM.
Input	Specified by the programmer.						
Output	Returned by the Presentation Manager* (PM) interface.						
Input/Output	Specified by the programmer and modified by PM.						

* Trademark of IBM Corporation

Return The return values are shown, together with possible errors, or TRUE/FALSE indicators if a Boolean function.

A list of possible errors (where appropriate) is included in this section. Some functions do not have error messages.

Note: Data types are given in C.

Remarks Additional information about the function, where required.

Related Functions Functions that can be used with the described function.

Example Code Example of how the function can be used.

Note: The functions in this book are named in mixed-case for readability, but are known to the system as uppercase character strings. For example, the function "GpiBeginArea" is actually the external name "GPIBEGINAREA."

If you are using a compiler that generates a mixed-case external name, you should code the OS/2[®] functions in uppercase.

Message Queues

For some functions, the Remarks section of the function description includes a statement that the function requires a message queue. This means that, before issuing the call, WinCreateMsgQueue must be issued by the same thread. For other functions, no previous WinCreateMsgQueue is required, and it is only necessary to issue WinInitialize from the same thread.

Error Severities

Each of the error conditions given in the list of errors for each call falls into one of these areas:

Warning The function detected a problem, but took some remedial action that enabled the function to complete successfully. The return code in this case indicates that the function completed successfully.

Error The function detected a problem for which it could not take any sensible remedial action. The system has recovered from the problem, and the state of the system with respect to the application remains the same as at the time when the function was requested. The system has not even partially executed the function (other than reporting the error).

Severe Error The function detected a problem from which the system could not reestablish its state, with respect to the application, at the time when that function was requested; that is, the system partially executed the function. This, therefore, necessitates the application performing some corrective activity to restore the system to some known state.

Unrecoverable Error The function detected some problem from which the system could not re-establish its state, with respect to the application, at the time when that call was issued. It is possible that the application cannot perform some corrective action to restore the system to some known state.

The WinGetLastError and WinGetErrorInfo functions can be used to find out more about an error (or warning) that occurs as a result of executing a call.

Header Files

All functions require an "include" for the system header file OS2.H:

```
#include <OS2.H>
```

Also, most functions require a "define" to select an appropriate (conditional) section of the header file, and hence, the required entry point. Where this is necessary, it is shown at the head of the function definition in the form:

```
#define INCL_name
```

Note: These "#defines" must precede the "#include <OS2.H>."

Helper Macros

A series of macros is defined for packing data into, and extracting data from, variables of MPARAM and MRESULT data types. They are used in conjunction with the WinSendMsg and the other message functions, and also inside window and dialog procedures.

These macros always cast their arguments to the specified type, so values of any of the types specified for each macro may be passed without additional casting. NULL may be used to pass unused parameter data.

Macros for packing data into a MPARAM variable:

```
/* Used to pass any pointer type: */
#define MPFROMP(p) ((MPARAM)(VOID*)(p))

/* Used to pass a window handle: */
#define MPFROMHWN(hwnd) ((MPARAM)(HWND)(hwnd))

/* Used to pass a CHAR, UCHAR, or BYTE: */
#define MPFROMCHAR(ch) ((MPARAM)(USHORT)(ch))

/* Used to pass a SHORT, USHORT, or BOOL: */
#define MPFROMSHORT(s) ((MPARAM)(USHORT)(s))

/* Used to pass two SHORTs or USHORTs: */
#define MPFROM2SHORT(s1, s2) ((MPARAM)MAKELONG(s1, s2))

/* Used to pass a SHORT and 2 UCHARs: (WM_CHAR msg)*/
#define MPFROMSH2CH(s, uch1, uch2)
((MPARAM)MAKELONG(s, MAKESHORT(uch1, uch2)))

/* Used to pass a LONG or ULONG: */
#define MPFROMLONG(l) ((MPARAM)(ULONG)(l))
```


Macros for extracting data from a MPARAM variable:

```
/* Used to get any pointer type: */
#define PVOIDFROMMP(mp)    ((VOID *) (mp))

/* Used to get a window handle: */
#define HWNDFROMMP(mp)     ((HWND) (mp))

/* Used to get CHAR, UCHAR, or BYTE: */
#define CHAR1FROMMP(mp)    ((UCHAR) (mp))
#define CHAR2FROMMP(mp)    ((UCHAR) ((ULONG) mp >> 8))
#define CHAR3FROMMP(mp)    ((UCHAR) ((ULONG) mp >> 16))
#define CHAR4FROMMP(mp)    ((UCHAR) ((ULONG) mp >> 24))

/* Used to get a SHORT, USHORT, or BOOL: */
#define SHORT1FROMMP(mp)    ((USHORT) ((ULONG) (mp)))
#define SHORT2FROMMP(mp)    ((USHORT) ((ULONG) mp >> 16))

/* Used to get a LONG or ULONG: */
#define LONGFROMMP(mp)      ((ULONG) (mp))
```

Macros for packing data into a MRESULT variable:

```
/* Used to pass any pointer type: */
#define MRFROMP(p)          ((MRESULT) (VOID *) (p))

/* Used to pass a SHORT, USHORT, or BOOL: */
#define MRFROMSHORT(s)      ((MRESULT) (USHORT) (s))

/* Used to pass two SHORTs or USHORTs: */
#define MRFROM2SHORT(s1, s2) ((MRESULT) MAKELONG(s1, s2))

/* Used to pass a LONG or ULONG: */
#define MRFROMLONG(l)       ((MRESULT) (ULONG) (l))
```

Macros for extracting data from a MRESULT variable:

```
/* Used to get any pointer type: */
#define PVOIDFROMMR(mr)     ((VOID *) (mr))

/* Used to get a SHORT, USHORT, or BOOL: */
#define SHORT1FROMMR(mr)    ((USHORT) ((ULONG) mr))
#define SHORT2FROMMR(mr)    ((USHORT) ((ULONG) mr >> 16))

/* Used to get a LONG or ULONG: */
#define LONGFROMMR(mr)      ((ULONG) (mr))
```

The following macros are for use with DDESTRUCT and DDEINIT structures:

```
/* Used to return a PSZ pointing to the DDE item name: */
#define DDES_PSZITEMNAME(pddes) \
    (((PSZ)pddes) + ((PDDESTRUCT)pddes)->offszItemName)

/* Used to return a PBYTE pointing to the DDE data: */
#define DDES_PABDATA(pddes) \
    (((PBYTE)pddes) + ((PDDESTRUCT)pddes)->offabData)

/* Used to convert a selector to a PDDESTRUCT: */
#define SELTOPDDES(sel) ((PDDESTRUCT)MAKEP(sel, 0))

/* Used to PDDESTRUCT to a selector for freeing / reallocating: */
#define PDDESTOSEL(pddes) (SELECTOROF(pddes))

/* Used to PDDEINIT to a selector for freeing: */
#define PDDEITOSEL(pddei) (SELECTOROF(pddei))
```

Addressing Elements in Arrays

Constants defining array elements are given values that are zero-based in C; that is, the numbering of the array elements starts at zero, not one.

For example, in the DevQueryCaps function, the sixth element of the *a/Array* parameter is CAPS_HEIGHT, which is equated to 5.

Count parameters related to such arrays always mean the actual number of elements available. Therefore, again using the DevQueryCaps function as an example, if all elements up to and including CAPS_HEIGHT are provided for, *lCount* could be set to (CAPS_HEIGHT + 1).

In functions for which the starting array element can be specified, this is always zero-based, and so the C element number constants can be used directly. For example, to start with the CAPS_HEIGHT element, the *lStart* parameter can be set to CAPS_HEIGHT.

Implicit Pointer Data Types

A data type name beginning with "P" (for example, PERRORCODE) is likely to be a pointer to another data type (in this instance, ERRORCODE).

In the data type summary, Appendix A, "Data Types" on page A-1, no explicit "typedefs" are shown for pointers. Therefore, if no data type definition can be found in the summary for a data type name "Pxxxxxx," it becomes a pointer to the data type "xxxxxx," for which a definition should be found in the summary.

The implicit type definition needed for such a pointer "Pxxxxxx" is:

```
typedef xxxxxx *Pxxxxxx;
```

Such definitions are provided by means of the system header file OS2.H.

Storage Mapping of Data Types

The storage mapping of the data types is dependent on the machine architecture. To be portable, applications must access the data types using the definitions supplied for that environment.

Double-Byte Character Set (DBCS)

Throughout this publication, you will see references to specific values for character strings. The values are for single-byte character set (SBCS). If you use the double-byte character set (DBCS), note that one DBCS character equals two SBCS characters.

Chapter 2. Device Functions

The following table shows all the Device (Dev) functions in alphabetic order.

C Name
DevCloseDC
DevEscape
DevOpenDC
DevPostDeviceModes
DevQueryCaps
DevQueryDeviceNames
DevQueryHardcopyCaps

DevCloseDC – Close Device Context

```
#define INCL_DEV /* Or use INCL_PM. Also in COMMON section */
```

HMF DevCloseDC (HDC hdc)

This function closes a device context.

Parameters

hdc (HDC) – input
Device-context handle.

Returns

Error indicator metafile handle (for a metafile device context)

DEV_ERROR Error occurred.

DEV_OK Device closed, but not a metafile device context.

Other Device closed, a metafile device context whose metafile handle is returned.

Possible returns from WinGetLastError

PMERR_NOT_CREATED_BY_DEVOPENDC An attempt has been made to destroy a device context using DevCloseDC that was not created using DevOpenDC.

PMERR_DC_IS_ASSOCIATED An attempt was made to associate a presentation space with a device context that was already associated or to destroy a device context that was associated.

PMERR_INV_HDC An invalid device-context handle or (micro presentation space) presentation-space handle was specified.

Remarks

If the device context is currently associated with a presentation space, or if it is created with the WinOpenWindowDC call (that is, it is a screen device context), an error is raised, and the device context is not closed.

If the device context being closed is a memory device context that has a bit map currently selected into it (see the GpiSetBitmap function), the bit map is automatically deselected before the device context is closed.

Any clip region currently in use for this device context is deleted.

Related Functions

Prerequisite Functions

- DevOpenDC

Other Related Functions

- WinOpenWindowDC

DevCloseDC — Close Device Context

Example Code

This example calls DevCloseDC to close a device context based on the handle returned from DevOpenDC.

```
#define INCL_DEV                /* Device Function definitions */
#include <os2.h>

HDC hdc;                       /* Device-context handle */
HMF hmf;                       /* error code (or metafile handle if
                               metafile device context) */

/* close the device context associated with handle hdc */
hmf = DevCloseDC(hdc);
```

DevEscape —

Escape

```
#define INCL_DEV /* Or use INCL_PM */
```

```
LONG DevEscape (HDC hdc, LONG ICode, LONG lInCount, PBYTE pbInData,  
                PLONG plOutCount, PBYTE pbOutData)
```

This function allows applications to access facilities of a device not otherwise available through the API. Escapes are, in general, sent to the presentation driver and must be understood by it.

Parameters

hdc (HDC) — input
Device-context handle.

ICode (LONG) — input
Escape code.

If the device context is of type OD_QUEUED with a PM_Q_STD spool file, some escapes are sent to the presentation driver and others are recorded in the spool file (depending on the escape code). If the device context is of type OD_METAFILE, all escapes are metafiled. If the device context is of any type other than OD_QUEUED (with a PM_Q_STD spool file) or OD_METAFILE, all escapes are sent to the presentation driver.

The description for each standard escape specifies which of these categories the escape falls into.

Devices can define additional escape functions using user *ICode* values, that have the following ranges:

32 768 through 40 959 Not metafiled and not recorded (sent to presentation driver for PM_Q_STD)

40 960 through 49 151 Metafiled only (sent to presentation driver for PM_Q_STD)

49 152 through 57 343 Metafiled and recorded (not sent to presentation driver) for PM_Q_STD

57 344 through 65 535 Recorded only (not sent to presentation driver for PM_Q_STD).

The following escapes are defined:

```
DEVESC_QUERYESCSUPPORT  
DEVESC_GETSCALINGFACTOR  
DEVESC_STARTDOC  
DEVESC_ENDDOC  
DEVESC_ABORTDOC  
DEVESC_NEWFRAME  
DEVESC_RAWDATA  
DEVESC_QUERYVIOCELLSIZES  
DEVESC_SETMODE
```

lInCount (LONG) — input
Input data count.

Number of bytes of data in the *pbInData* buffer.

pbInData (PBYTE) — input
The input data required for this escape.

plOutCount (PLONG) — input/output
Output data count.

plOutCount is the number of bytes of data in the *pbOutData* buffer.

If data is returned in *pbOutData*, *plOutCount* is updated to the number of bytes of data returned.

DevEscape – Escape

pbOutData (PBYTE) – output
Output data.

pbOutData is a buffer that receives the output from this escape. If *plOutCount* is null, no data is returned.

Returns

Implementation error indicator:

DEVESC_ERROR	Error
DEVESC_NOTIMPLEMENTED	Escape not implemented for specified code
DEV_OK	OK.

Possible returns from WinGetLastError

PMERR_INV_ESCAPE_CODE	An invalid code parameter was specified with DevEscape.
PMERR_INV_HDC	An invalid device-context handle or (micro presentation space) presentation-space handle was specified.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_ESC_CODE_NOT_SUPPORTED	The code specified with DevEscape is not supported by the target device-driver.
PMERR_INV_ESCAPE_DATA	An invalid data parameter was specified with DevEscape.

Remarks

The data fields for standard escapes are:

DEVESC_QUERYESCSUPPORT	<p>Queries whether a particular escape is implemented by the presentation driver. The return value gives the result.</p> <p>This escape is not metafiled or recorded.</p> <p><i>lInCount</i> Number of bytes pointed to by <i>pbInData</i>.</p> <p><i>pbInData</i> The buffer contains an escape code value specifying the escape function to be checked.</p> <p><i>plOutCount</i> Not used; can be set to 0.</p> <p><i>pbOutData</i> Not used; can be set to null.</p>
DEVESC_GETSCALINGFACTOR	<p>Returns the scaling factors for the x and y axes of a printing device. For each scaling factor, an exponent of two is put in <i>pbOutData</i>. Thus, the value 3 is used if the scaling factor is 8.</p> <p>Scaling factors are used by devices that cannot support graphics at the same resolution as the device resolution.</p> <p>This escape is not metafiled or recorded.</p> <p><i>lInCount</i> Not used; can be set to 0.</p> <p><i>pbInData</i> Not used; can be set to null.</p> <p><i>plOutCount</i> The number of bytes of data pointed to by <i>pbOutData</i>. On return, this is updated to the number of bytes returned.</p> <p><i>pbOutData</i> The address of a SFACTORS structure, which on return contains the scaling factors for the x and y axes.</p>
DEVESC_STARTDOC	<p>Indicates that a new print job is starting. All subsequent output to the device context is spooled under the same job identifier until a DEVESC_ENDDOC occurs.</p> <p>A GpiAssociate function must be issued to associate the presentation space with the device context before issuing this escape.</p>

DevEscape — Escape

This escape is metafiled but not recorded.

lInCount Number of bytes pointed to by *pbInData*.

pbInData The buffer contains a null-terminated string, specifying the name of the document.

plOutCount Not used; can be set to 0.

pbOutData Not used; can be set to null.

DEVESC_ENDDOC

Ends a print job started by DEVESC_STARTDOC.

This escape is metafiled but not recorded.

lInCount Not used; can be set to 0.

pbInData Not used; can be set to null.

plOutCount Set equal to 2.

pbOutData The buffer contains a USHORT specifying the job identifier if a spooler print job is created.

DEVESC_ABORTDOC

Aborts the current job, erasing everything the application has written to the device since the last DEVESC_STARTDOC, including the DEVESC_STARTDOC.

This escape is metafiled but not recorded.

lInCount Not used; can be set to 0

pbInData Not used; can be set to null

plOutCount Not used; can be set to 0

pbOutData Not used; can be set to null.

DEVESC_NEWFRAME

Signals when an application has finished writing to a page and wants to start a new page. It is similar to GpiErase processing for a screen device context, and causes a reset of the attributes.

This escape is used with a printer device to advance to a new page.

This escape is metafiled and recorded.

lInCount Not used; can be set to 0

pbInData Not used; can be set to null

plOutCount Not used; can be set to 0

pbOutData Not used; can be set to null.

DEVESC_RAWDATA

Allows an application to send data directly to a presentation driver. For example, in the case of a printer driver, this could be a printer data stream.

If DEVESC_RAWDATA is mixed with other data (such as GPI data) being sent to the same page of a device context, the results are unpredictable and depend upon the action taken by the presentation driver. For example, a presentation driver might ignore GPI data if DEVESC_RAWDATA is mixed with it on the same page. In general, DEVESC_RAWDATA should be sent either to a separate page (using the DEVESC_NEWFRAME escape to obtain a new page) or to a separate document (using the DEVESC_STARTDOC and DEVESC_ENDDOC escapes to create a new document).

This escape is metafiled and recorded.

lInCount Number of bytes pointed to by *pbInData*

pbInData Pointer to the raw data

plOutCount Not used; can be set to 0

pbOutData Not used; can be set to null.

DEVESC_QUERYVIOCELLSIZES

Returns the VIO cell sizes supported by the presentation driver.

This escape is not metafiled or recorded.

lInCount Not used; can be set to 0

pbInData Not used; can be set to null.

plOutCount The number of bytes of data pointed to by *pbOutData*. It must be an even multiple of the size in bytes of the LONG data type. On return, this is updated to the number of bytes returned.

pbOutData The address of a buffer, which on return contains a VIOSIZECOUNT structure, immediately followed by *count* copies of a VIOFONTCELLSIZE structure.

If *plOutCount* is less than the size of a LONG data type, *plOutCount* is updated to zero, and nothing is returned in the buffer pointed to by *pbOutData*.

If *plOutCount* is equal to the size of a LONG data type, *pbOutData* returns the number of VIO cell sizes that can be returned by this escape. The buffer pointed to by *pbOutData* is updated so that *maxcount* is the number of VIO cell sizes that can be returned.

If *plOutCount* is greater than the size of a LONG data type, *pbOutData* returns the VIO cell sizes that are supported. The buffer pointed to by *pbOutData* is updated so that:

- *maxcount* is the number of VIO cell sizes that can be returned
- *count* is the number of VIO cell sizes returned (may be zero if *plOutCount* is equal to twice the size of a LONG data type)
- *count* copies of a VIOFONTCELLSIZE structure are returned.

DEVESC_SETMODE

Sets the printer into a particular mode. It is optional for printer drivers to support this escape, but those that do support it need to be aware of the code page of any built-in fonts. For example, if only code page 437 is built in, it is used if 437 is requested by DEVESC_SETMODE. However, if code page 865 is requested, a suitable code page/font could be downloaded.

This escape is metafiled and recorded.

lInCount Number of bytes pointed to by *pbInData*

pbInData Buffer contains an ESCSETMODE structure

plOutCount Not used; can be set to 0

pbOutData Not used; can be set to null.

Related Functions

Prerequisite Functions

- DevOpenDC

Other Related Functions

- GpiAssociate(for DEVESC_STARTDOC)
- GpiErase(for DEVESC_NEWFRAME)

DevEscape —

Escape

Graphic Elements and Orders

DevEscape functions generate orders only when metafilling.

Order: **Extended Escape**

Example Code

This example uses DevEscape to access facilities of a device that would otherwise be unavailable through the normal Device API set. Here, a new page in a print job is started.

```
#define INCL_DEV      /* Device Function definitions */
#include <os2.h>

LONG lResult;        /* Error code or not implemented      */
                        warning code                          */
HDC  hdc;            /* Device-context handle      */
LONG plOutCount;     /* length of output buffer(input),
                        number of bytes returned(output) */
PBYTE pbOutData;     /* output buffer              */

/* for the NEWFRAME, input and output buffers are not used,
   so set the buffer lengths to zero(0) and set the buffers to
   NULL */
plOutCount = 0;
pbOutData = NULL;

lResult = DevEscape(hdc, DEVEVC_NEWFRAME, 0L, NULL, &plOutCount,
                    pbOutData);
```

DevOpenDC – Open Device Context

```
#define INCL_DEV /* Or use INCL_PM. Also in COMMON section */
```

**HDC DevOpenDC (HAB hab, LONG IType, PSZ pszToken, LONG ICount,
PDEVOPENDATA pdopData, HDC hdcComp)**

This function creates a device context.

Parameters

hab (HAB) – input
Anchor-block handle.

IType (LONG) – input
Type of device context:

OD_QUEUED	<p>A device, such as a printer or plotter, for which output is to be queued.</p> <p>Certain restrictions apply for this device type; see “Metafile Restrictions” on page G-1.</p>
OD_DIRECT	<p>A device, such as a printer or plotter, for which output is not to be queued.</p>
OD_INFO	<p>A device, such as a printer or plotter, but the device context is used only to retrieve information (for example, font metrics). Drawing can be performed to a presentation space associated with such a device context, but no output medium is updated.</p>
OD_METAFILE	<p>The device context is used to write a metafile. The presentation page defines the area of interest within the picture in the metafile. See OD_METAFILE_NOQUERY.</p> <p>Certain restrictions apply for this device type; see “Metafile Restrictions” on page G-1.</p>
OD_METAFILE_NOQUERY	<p>The device context is used to write a metafile.</p> <p>Functionally, this device type is the same as OD_METAFILE, except that querying of attributes is not allowed with a presentation space while it is associated with an OD_METAFILE_NOQUERY device context. If querying of attributes is not required, OD_METAFILE_NOQUERY should be used in preference to OD_METAFILE, since it gives improved performance.</p> <p>Certain restrictions apply for this device type; see “Metafile Restrictions” on page G-1.</p>
OD_MEMORY	<p>A device context that is used to contain a bit map. The <i>hdcComp</i> parameter identifies a device with which the memory device context is to be compatible.</p>

pszToken (PSZ) – input
Device-information token.

This identifies the device information, held in the initialization file. This information is the same as that which may be pointed to by *pdopData*; any information that is obtained from *pdopData* overrides the information obtained by using this parameter.

If *pszToken* is specified as “*”, no device information is taken from the initialization file.

OS/2 behaves as if “*” is specified, but it allows any string.

DevOpenDC —

Open Device Context

ICount (LONG) — input
Number of items.

This is the number of items present in the *pdopData* parameter. This can be less than the full list if omitted items are irrelevant, or are supplied from *pszToken* or elsewhere.

pdopData (PDEVOPENDATA) — input
Open-device-context data area.

hdcComp (HDC) — input
Compatible-device-context handle.

When *IType* is *OD_MEMORY*, this parameter is a handle to a device context compatible with bit maps that are to be used with this device context.

If *hdcComp* is *NULLHANDLE*, compatibility with the screen is assumed.

Returns

Device-context handle:

DEV_ERROR Error

≠0 Device-context handle.

Possible returns from *WinGetLastError*

PMERR_INV_DC_TYPE	An invalid type parameter was specified with <i>DevOpenDC</i> , or a function was issued that is invalid for a <i>OD_METAFILE_NOQUERY</i> device context.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_INV_DC_DATA	An invalid data parameter was specified with <i>DevOpenDC</i> .
PMERR_INV_HDC	An invalid device-context handle or (micro presentation space) presentation-space handle was specified.
PMERR_INV_DRIVER_NAME	A driver name was specified which has not been installed.
PMERR_INV_LOGICAL_ADDRESS	An invalid device logical address was specified.

Remarks

A device context is a means of writing to a particular device. Before using GPI functions to cause output to be directed to the device context, the *GpiAssociate* function call must be issued (or the *GPIA_ASSOC* option specified on *GpiCreatePS*).

DevOpenDC cannot be used to open a device context for a screen window; use *WinOpenWindowDC* instead.

The device context is owned by the process from which *DevOpenDC* is issued. It cannot be accessed directly from any other process. If it still exists when the process terminates, it is automatically deleted by the system. When using a device context type of *OD_METAFILE_NOQUERY* the querying of attributes is not allowed. To improve performance of this type of metafile no error checking is performed to ensure that such API calls are not attempted. Query calls are accepted but the results returned are undefined.

This function requires the existence of a message queue.

Related Functions

Prerequisite Functions

- WinInitialize

Other Related Functions

- DevCloseDC
- GpiAssociate(for the output of GPI data)
- PrfQueryProfileString
- WinOpenWindowDC
- WinQueryWindow

Example Code

This example calls DevOpenDC to create a memory device context with screen compatibility and then associates that context with a newly created presentation space.

```
#define INCL_DEV                /* Device Function definitions */
#define INCL_GPICONTROL        /* GPI control Functions */
#include <os2.h>

HDC  hdc;                      /* Device-context handle */
HAB  hab;                      /* Anchor-block handle */
/* context data structure */
DEVOPENSTRUC dop = {NULL, "DISPLAY", NULL, NULL, NULL, NULL,
                    NULL, NULL, NULL};

HPS  hps;                      /* presentation-space handle */
SIZEL sizl={0, 0};            /* use same page size as device */

/* create memory device context */
hdc = DevOpenDC(hab, OD_MEMORY, "", 5L, (PDEVOPENDATA)&dop, NULLHANDLE);

/* create a presentation space associated with the context */
hps = GpiCreatePS(hab, hdc, &sizl, GPIA_ASSOC | PU_PELS);
```

DevPostDeviceModes —

Post Device Modes

```
#define INCL_DEV /* Or use INCL_PM */
```

LONG DevPostDeviceModes (HAB hab, PDRIVDATA pdrvDriverData, PSZ pszDriverName, PSZ pszDeviceName, PSZ pszName, ULONG flOptions)

This function returns, and optionally sets job properties.

Parameters

hab (HAB) — input
Anchor-block handle.

pdrvDriverData (PDRIVDATA) — input/output
Driver data.

A data area that, on return, contains device data defined by the presentation driver. If the pointer to the area is NULL, this function returns the required size of the data area.

The format of the data is the same as that which occurs within the DEVOPENSTRUC structure, passed on the *pdrvData* parameter of DevOpenDC.

pszDriverName (PSZ) — input
Device-driver name. A string containing the name of the presentation driver; for example, "LASERJET."

pszDeviceName (PSZ) — input
Device-type name.

Null-terminated string in a 32-byte field, identifying the device type; for example, "HP LaserJet IID" (model number). Valid names are defined by device drivers.

Note: This parameter always overrides the data in the *szDeviceName*[32] field of the DRIVDATA structure, passed in the *pdrvDriverData* parameter.

pszName (PSZ) — input
Device name.

A name that identifies the device; for example, "PRINTER1." If DPDM_POSTJOBPROP is specified in the *flOptions* parameter, the *pszName* parameter can be NULL.

flOptions (ULONG) — input
Dialog options.

Options that control whether a dialog is displayed.

DPDM_POSTJOBPROP

This function allows the user to set properties for the print job by displaying a dialog and returning the updated job properties. Examples of job properties are paper size, paper orientation, and single-sided or duplex.

The printer is configured in the shell using a dialog provided by the presentation driver. The configuration describes the actual printer setup such as number of paper bins, available paper sizes, and any installed hardware fonts.

Before the job properties dialog is displayed the presentation driver merges any changes in the printer configuration with the data passed in the *pdrvDriverData* parameter. This allows, for example new paper sizes to be added into the job properties dialog. The parameter *pszName* can be specified as NULL although this is not recommended because the presentation driver cannot easily find the printer configuration to merge.

DevPostDeviceModes – Post Device Modes

It is the responsibility of the application to retrieve and store job properties. An application can choose to store job properties either on a per document or per application basis. The job properties can then be passed into DevOpenDC. Initial (default) job properties can be retrieved using DPDM_QUERYJOBPROP option.

The application cannot tell if the user modified the job properties or just cancelled the dialog. Hence the job properties returned in the *pdrivDriverData* parameter must always be stored.

The shell allows users to specify default job properties for a printer. The spooler API SplQueryQueue can be used to retrieve these defaults. The spooler automatically adds the default job properties for a printer to any jobs that are submitted without job properties.

DPDM_QUERYJOBPROP

Do not display a dialog. Return the default job properties. These defaults are derived from the defaults for the chosen device; for example, "HP Laserjet IID" and the printer setup specified via the shell printer driver configuration dialog.

Returns

Size/error indicator.

Value depends on what was passed as the pointer to *pdrivDriverData*:

NULL

DPDM_ERROR Error

DPDM_NONE No settable options

>0 Size in bytes required for *pdrivDriverData*.

Other

DPDM_ERROR Error

DPDM_NONE No settable options

DEV_OK OK.

Possible returns from WinGetLastError

PMERR_INV_DRIVER_DATA

Invalid driver data was specified.

PMERR_DRIVER_NOT_FOUND

The device driver specified with DevPostDeviceModes was not found.

PMERR_INV_DEVICE_NAME

An invalid devicename parameter was specified with DevPostDeviceModes.

PMERR_INV_LOGICAL_ADDRESS

An invalid device logical address was specified.

Remarks

An application can first call this function with a NULL data pointer to find out how much storage is needed for the data area. Having allocated the storage, the application can then make the call a second time for the data to be entered. The returned data can then be passed in DevOpenDC as *pdrivDriverData* within the *pdopData* parameter.

Calling this function requires the existence of a message queue.

Use SplEnumDevice or SplEnumPrinter with *fType* set to SPL_PR_DIRECT_DEVICE or SPL_PR_QUEUEUED_DEVICE to get a list of all the devices.

To get information about a specific device use SplQueryDevice.

DevPostDeviceModes —

Post Device Modes

Related Functions

- DevOpenDC

Example Code

This example shows how to call DevPostDeviceModes and allocate a new buffer, if necessary, for the larger job properties (DRIVDATA structure).

```
#define INCL_DEV
#define INCL_DOS
#include <os2.h>
#include <memory.h>

{
    ULONG          devrc=FALSE;
    HAB            hab;
    PSZ            pszPrinter;
    HDC            hdc=NULL;
    PDRIVDATA      pOldDrivData;
    PDRIVDATA      pNewDrivData=NULL;
    PDEVOPENSTRUC dops;
    LONG           buflen;

    /* check size of buffer required for job properties */
    buflen = DevPostDeviceModes( hab,
                                NULL,
                                dops->pszDriverName,
                                dops->pdriv->szDeviceName,
                                pszPrinter,
                                DPDM_POSTJOBPROP
                                );

    /* return error to caller */
    if (buflen<=0)
        return(buflen);

    /* allocate some memory for larger job properties and */
    /* return error to caller */

    if (buflen != dops->pdriv->cb)
    {
        if (DosAllocMem((PPVOID)&pNewDrivData,buflen,fALLOC))
            return(DPDM_ERROR);
    }

    /* copy over old data so driver can use old job */
    /* properties as base for job properties dialog */
    pOldDrivData = dops->pdriv;
    dops->pdriv = pNewDrivData;
    memcpy( (PSZ)pNewDrivData, (PSZ)pOldDrivData, pOldDrivData->cb );

    /* display job properties dialog and get updated */
    /* job properties from driver */

    devrc = DevPostDeviceModes( hab,
                                dops->pdriv,
                                dops->pszDriverName,
                                dops->pdriv->szDeviceName,
                                pszPrinter,
                                DPDM_POSTJOBPROP
                                );

    return(devrc);
}
```

DevQueryCaps – Query Device Capabilities

```
#define INCL_DEV /* Or use INCL_PM. Also in COMMON section */
```

```
BOOL DevQueryCaps (HDC hdc, LONG IStart, LONG ICount, PLONG alArray)
```

This function queries the device characteristics.

Parameters

hdc (HDC) – input
Device-context handle.

IStart (LONG) – input
First item of information.

The number of the first item of information to be returned in *alArray*, counting from zero.

ICount (LONG) – input
Count of items of information.

This is the count to be returned in *alArray*. It must be greater than zero.

alArray (PLONG) – output
Device capabilities.

Array of *ICount* elements, starting with *IStart*. The array elements are numbered consecutively, starting with **CAPS_FAMILY**. The element number constants start with 0. See the appropriate bindings reference.

If *IStart + ICount – 1* exceeds the current highest-defined element number, elements beyond the highest are returned as 0.

CAPS_FAMILY

Device type (values as for *IType* in DevOpenDC).

CAPS_IO_CAPS

Device input/output capability:

CAPS_IO_DUMMY

Dummy device

CAPS_SUPPORTS_OP

Device supports output

CAPS_SUPPORTS_IP

Device supports input

CAPS_SUPPORTS_IO

Device supports output and input.

CAPS_TECHNOLOGY

Technology:

CAPS_TECH_UNKNOWN

Unknown

CAPS_TECH_VECTOR_PLOTTER

Vector plotter

CAPS_TECH_RASTER_DISPLAY

Raster display

CAPS_TECH_RASTER_PRINTER

Raster printer

CAPS_TECH_RASTER_CAMERA

Raster camera

CAPS_TECH_POSTSCRIPT

PostScript device.

DevQueryCaps — Query Device Capabilities

CAPS_DRIVER_VERSION	Version identifier of the presentation driver. The high order word of the version identifier is 0. The low order word identifies the release, for example 0x0120 is release 1.2.
CAPS_WIDTH	Media width (for a full screen, maximized window for displays) in pels.
CAPS_HEIGHT	Media depth (for a full screen, maximized window for displays) in pels. (For a plotter, a pel is defined as the smallest possible displacement of the pen and can be smaller than a pen width.)
CAPS_WIDTH_IN_CHARS	Media width (for a full screen, maximized window for displays) in default character columns.
CAPS_HEIGHT_IN_CHARS	Media depth (for a full screen, maximized window for displays) in default character rows.
CAPS_HORIZONTAL_RESOLUTION	Horizontal resolution of device in pels per meter.
CAPS_VERTICAL_RESOLUTION	Vertical resolution of device in pels per meter.
CAPS_CHAR_WIDTH	Default character-box width in pels for VIO.
CAPS_CHAR_HEIGHT	Default character-box height in pels for VIO.
CAPS_SMALL_CHAR_WIDTH	Default small-character box width in pels for VIO. This is 0 if there is only one character-box size.
CAPS_SMALL_CHAR_HEIGHT	Default small-character box height in pels for VIO. This is 0 if there is only one character-box size.
CAPS_COLORS	Number of distinct colors supported at the same time, including reset (gray scales count as distinct colors). If loadable color tables are supported, this is the number of entries in the device color table. For plotters, the value returned is the number of pens plus one (for the background).
CAPS_COLOR_PLANES	Number of color planes.
CAPS_COLOR_BITCOUNT	Number of adjacent color bits for each pel (within one plane).
CAPS_COLOR_TABLE_SUPPORT	Loadable color table support: CAPS_COLTABL_RGB_8 1 if RGB color table can be loaded, with a minimum support of 8 bits each for red, green, and blue. CAPS_COLTABL_RGB_8_PLUS 1 if color table with other than 8 bits for each primary color can be loaded. CAPS_COLTABL_TRUE_MIX 1 if true mixing occurs when the logical color table has been realized, providing that the size of the logical color table is not greater than the number of distinct colors supported (see element CAPS_COLORS). CAPS_COLTABL_REALIZE 1 if a loaded color table can be realized.
CAPS_MOUSE_BUTTONS	The number of pointing device buttons that are available. A returned value of 0 indicates that there are no pointing device buttons available.

DevQueryCaps —

Query Device Capabilities

CAPS_FOREGROUND_MIX_SUPPORT Foreground mix support:

CAPS_FM_OR

Logical OR.

CAPS_FM_OVERPAINT

Overpaint.

CAPS_FM_XOR

Logical XOR.

CAPS_FM_LEAVEALONE

Leave alone.

CAPS_FM_AND

Logical AND.

CAPS_FM_GENERAL_BOOLEAN

All other mix modes; see GpiSetMix.

The value returned is the sum of the values appropriate to the mixes supported. A device capable of supporting OR must, as a minimum, return CAPS_FM_OR + CAPS_FM_OVERPAINT + CAPS_FM_LEAVEALONE, signifying support for the mandatory mixes OR, overpaint, and leave-alone.

Note that these numbers correspond to the decimal representation of a bit string that is six bits long, with each bit set to 1 if the appropriate mode is supported.

Those mixes returned as supported are guaranteed for all primitive types. For more information, see GpiSetMix.

CAPS_BACKGROUND_MIX_SUPPORT Background mix support:

CAPS_BM_OR

Logical OR.

CAPS_BM_OVERPAINT

Overpaint.

CAPS_BM_XOR

Logical XOR.

CAPS_BM_LEAVEALONE

Leave alone.

CAPS_BM_AND

Logical AND.

CAPS_BM_GENERAL_BOOLEAN

All other mix modes; see GpiSetBackMix.

The value returned is the sum of the values appropriate to the mixes supported. A device must, as a minimum, return CAPS_BM_OVERPAINT + CAPS_BM_LEAVEALONE, signifying support for the mandatory background mixes overpaint, and leave-alone.

Note that these numbers correspond to the decimal representation of a bit string that is four bits long, with each bit set to 1 if the appropriate mode is supported.

Those mixes returned as supported are guaranteed for all primitive types. For more information, see GpiSetBackMix.

CAPS_VIO_LOADABLE_FONTS

Number of fonts that can be loaded for VIO.

DevQueryCaps – Query Device Capabilities

CAPS_WINDOW_BYTE_ALIGNMENT	Whether or not the client area of VIO windows should be byte-aligned:
	CAPS_BYTE_ALIGN_REQUIRED Must be byte-aligned.
	CAPS_BYTE_ALIGN_RECOMMENDED More efficient if byte-aligned, but not required.
	CAPS_BYTE_ALIGN_NOT_REQUIRED Does not matter whether byte-aligned.
CAPS_BITMAP_FORMATS	Number of bit-map formats supported by device.
CAPS_RASTER_CAPS	Capability for device raster operations:
	CAPS_RASTER_BITBLT 1 if GpiBitBlt and GpiWCBitBlt supported
	CAPS_RASTER_BANDING 1 if banding is supported
	CAPS_RASTER_BITBLT_SCALING 1 if GpiBitBlt and GpiWCBitBlt with scaling supported.
	CAPS_RASTER_SET_PEL 1 if GpiSetPel supported.
	CAPS_RASTER_FONTS 1 if this device can draw raster fonts.
	CAPS_RASTER_FLOOD_FILL 1 if GpiFloodFill is supported.
CAPS_MARKER_HEIGHT	Default marker-box height in pels.
CAPS_MARKER_WIDTH	Default marker-box width in pels.
CAPS_DEVICE_FONTS	Number of device-specific fonts.
CAPS_GRAPHICS_SUBSET	Graphics drawing subset supported. (3 indicates GOCA DR/3)
CAPS_GRAPHICS_VERSION	Graphics architecture version number supported. (1 indicates Version 1)
CAPS_GRAPHICS_VECTOR_SUBSET	Graphics vector drawing subset supported. (2 indicates GOCA VS/2)
CAPS_DEVICE_WINDOWING	Device windowing support:
	CAPS_DEV_WINDOWING_SUPPORT 1 if device supports windowing. Other bits are reserved 0.
CAPS_ADDITIONAL_GRAPHICS	Additional graphics support:
	CAPS_GRAPHICS_KERNING_SUPPORT 1 if device supports kerning.
	CAPS_FONT_OUTLINE_DEFAULT 1 if device has a default outline font.
	CAPS_FONT_IMAGE_DEFAULT 1 if device has a default image font.
	CAPS_SCALED_DEFAULT_MARKERS 1 if default markers are to be scaled by the marker-box attribute.
	CAPS_COLOR_CURSOR_SUPPORT 1 if device supports colored cursors.
	CAPS_PALETTE_MANAGER 1 if device supports palette functions (see GpiCreatePalette).

DevQueryCaps – Query Device Capabilities

CAPS_COSMETIC_WIDELINE_SUPPORT

1 if device supports cosmetic thick lines (see GpiSetLineWidth).

CAPS_ENHANCED_TEXT

1 if device supports full font file description and text alignment.

Other bits are reserved 0.

CAPS_PHYS_COLORS

Maximum number of distinct colors available on the device.

CAPS_COLOR_INDEX

Maximum logical color-table index supported for this device. For the EGA and VGA drivers, the value is 63.

CAPS_GRAPHICS_CHAR_WIDTH

Default graphics character-box width, in pels.

CAPS_GRAPHICS_CHAR_HEIGHT

Default graphics character-box height, in pels.

CAPS_HORIZONTAL_FONT_RES

Effective horizontal device resolution in pels per inch, for the purpose of selecting fonts.

For printers, this is the actual device resolution, but for displays it may differ from the actual resolution for reasons of legibility.

CAPS_VERTICAL_FONT_RES

Effective vertical device resolution in pels per inch, for the purpose of selecting fonts.

CAPS_DEVICE_FONT_SIM

Identifies which simulations are valid on device fonts.

Valid flags are:

CAPS_DEVICE_FONT_SIM_BOLD
CAPS_DEVICE_FONT_SIM_ITALIC
CAPS_DEVICE_FONT_SIM_UNDERSCORE
CAPS_DEVICE_FONT_SIM_STRIKEOUT

CAPS_LINEWIDTH_THICK

Cosmetic thickness of lines and arcs on this device, when *fxLineWidth* is LINEWIDTH_THICK (see GpiSetLineWidth). The units are pels. A value of 0 is interpreted as 2 pels.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HDC

An invalid device-context handle or (micro presentation space) presentation-space handle was specified.

PMERR_INV_QUERY_ELEMENT_NO

An invalid start parameter was specified with DevQueryCaps.

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

DevQueryCaps — Query Device Capabilities

Remarks

GpiQueryDevice can be used to find the handle of the currently associated device context.

Related Functions

Prerequisite Functions

- DevOpenDC(for CAPS_FAMILY)

Other Related Functions

- DevQueryDeviceNames
- DevQueryHardcopyCaps
- GpiQueryDevice
- GpiSetMix(for CAPS_FOREGROUND_MIX_SUPPORT)
- GpiSetBackMix(for CAPS_BACKGROUND_MIX_SUPPORT)

Example Code

In this example the driver is queried to see if it supports input, output, or both. Note that a valid device context handle must be passed. This example assumes a DevOpenDC call has been made to obtain the device context handle.

```
#define INCL_DEV
#include <OS2.H>

HDC hdc;
LONG lStart;
LONG lCount;
BOOL flreturn;
LONG a1Array[CAPS_TECHNOLOGY];
lCount = CAPS_TECHNOLOGY;
lStart = CAPS_FAMILY;

flreturn = DevQueryCaps(hdc,          /* device context handle */
                        lStart,        /* number of first item */
                        lCount,        /* count of items */
                        a1Array);      /* array of longs which */
                                      /* will contain the return */
                                      /* information. */

switch(a1Array[CAPS_IO_CAPS])        /* we test the CAPS_IO_CAPS */
{                                     /* element of the array to */
    case CAPS_IO_SUPPORTS_OP:        /* find out which options */
                                      /* are supported. */
                                      /* device supports output.*/
        break;
    case CAPS_IO_SUPPORTS_IP:        /* device supports input. */
        break;
    case CAPS_IO_SUPPORTS_IO:        /* device supports both */
                                      /* input and output. */
        break;
    default:
        break;
}
```

DevQueryDeviceNames – Query Device Names

```
#define INCL_DEV /* Or use INCL_PM */
```

BOOL DevQueryDeviceNames (HAB hab, PSZ pszDriverName, PLONG pldn,
PSTR32 aDeviceName, PSTR64 aDeviceDesc, PLONG pldt,
PSTR16 aDataType)

This function causes a presentation driver to return the names, descriptions, and data types of the devices it supports.

Parameters

hab (HAB) – input

Anchor-block handle.

pszDriverName (PSZ) – input

Fully-qualified name of the file containing the presentation driver.

The file-name extension is DRV.

pldn (PLONG) – input/output

Maximum number of device names and descriptions that can be returned.

pldn can have the following values:

Zero The number of device names and descriptions supported is returned; *aDeviceName* and *aDeviceDesc* are not updated.

Nonzero *pldn* is updated to the number returned in *aDeviceName* and *aDeviceDesc*; *aDeviceName* and *aDeviceDesc* are updated.

aDeviceName (PSTR32) – output

Device-name array.

An array of null-terminated strings, each element of which identifies a particular device. Valid names are defined by presentation drivers.

aDeviceDesc (PSTR64) – output

Device-description array.

An array of null-terminated strings, each element of which is a description of a particular device. Valid descriptions are defined by presentation drivers.

pldt (PLONG) – input/output

Maximum number of data types that can be returned.

pldt can have the following values:

Zero The number of data types supported is returned, and *aDataType* is not updated.

Nonzero *pldt* is updated to the number returned, and *aDataType* is updated.

aDataType (PSTR16) – output

Data type array.

An array of null-terminated strings, each element of which identifies a data type. Valid data types are defined by presentation drivers.

DevQueryDeviceNames —

Query Device Names

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_LENGTH_OR_COUNT An invalid length or count parameter was specified.

Remarks

An application can first call this function with *pldn* and *pldt* set to 0 to find how much storage is needed for the data areas. Having allocated the storage, the application calls the function a second time for the data to be entered.

'HP LaserJet'' II' is an example of a device name, 'HP LaserJet II' is an example of a device description, and 'PM_Q_STD' is an example of a data type.

Related Functions

- DevQueryCaps
- DevQueryHardcopyCaps

DevQueryDeviceNames – Query Device Names

Example Code

This example uses DevQueryDeviceNames to return the names, descriptions, and data types of supported devices for a presentation driver. The first call to DevQueryDeviceNames determines the number of names, description, and data types available; after allocating the arrays, the second call actually returns the information in the arrays.

```
#define INCL_DEV                /* Device Function definitions */
#define INCL_DOSMEMMGR          /* DOS Memory Manager Functions */
#include <os2.h>

BOOL fSuccess;                /* success indicator */
HAB hab;                      /* Anchor-block handle */
LONG pldn = 0L;               /* number of device names/descriptions */
LONG pldt = 0L;               /* number of data types */
PSTR32 aDeviceName;           /* array of device names */
PSTR64 aDeviceDesc;           /* array of device descriptions */
PSTR16 aDataType;             /* array of data types */

/* query number of supported names/descriptions/data types
   (pldn & pldt both 0) */
fSuccess = DevQueryDeviceNames(hab, "IBM4201.DRV", &pldn,
                               aDeviceName, aDeviceDesc, &pldt,
                               aDataType);

if (fSuccess)
{
    /* allocate arrays */
    DosAllocMem((VOID *)aDeviceName, (ULONG)pldn*sizeof(STR32),
                PAG_COMMIT | PAG_WRITE);
    DosAllocMem((VOID *)aDeviceDesc, (ULONG)pldn*sizeof(STR64),
                PAG_COMMIT | PAG_WRITE);
    DosAllocMem((VOID *)aDataType, (ULONG)pldt*sizeof(STR16),
                PAG_COMMIT | PAG_WRITE);

    /* query supported device information */
    fSuccess = DevQueryDeviceNames(hab, "IBM4201.DRV", &pldn,
                                   aDeviceName, aDeviceDesc, &pldt,
                                   aDataType);
}
```

DevQueryHardcopyCaps – Query Hardcopy Caps

```
#define INCL_DEV /* Or use INCL_PM */
```

```
LONG DevQueryHardcopyCaps (HDC hdc, LONG IStartForm, LONG IForms,  
                           PHCIHcInfo)
```

This function queries the hard-copy capabilities of a device.

Parameters

hdc (HDC) – input
Device-context handle.

IStartForm (LONG) – input
Start-forms code.

Forms-code number from which the query is to start. The first forms code has the value 0. *IStartForm* is used with *IForms*.

IForms (LONG) – input
Number of forms to query.

If 0, the number of forms codes defined is returned. If greater than zero, this function returns the number of forms codes for which information is returned.

For example, if there are five forms codes defined, and *IStartForm* = 2 and *IForms* = 3, a query is performed for forms codes 2, 3, and 4. The result is returned in the buffer pointed to by *phciHcInfo*.

phciHcInfo (PHCIHcInfo) – output
Hard-copy capabilities information.

A buffer containing the results of the query. The result consists of *IForms* copies of the HCINFO structure.

At least one of the defined forms codes must have the HCAPS_CURRENT bit set. There might be more than one with either the HCAPS_CURRENT or the HCAPS_SELECTABLE bits set.

For a job to be selected by the spooler for printing, each one of the forms specified in the FORM spooler parameter (see *pszSpoolerParams* in DEVOPENSTRUC) must be either HCAPS_CURRENT or HCAPS_SELECTABLE. The following are possibilities:

- All forms specified are HCAPS_SELECTABLE.
- The single form specified is HCAPS_CURRENT.
- One of the forms is HCAPS_CURRENT, and all of the others are HCAPS_SELECTABLE.

Returns

Details of forms:

DQHC_ERROR Error.

≥0 If *IForms* equals 0, number of forms available.
If *IForms* does not equal 0, number of forms returned.

Possible returns from WinGetLastError

PMERR_INV_HDC An invalid device-context handle or (micro presentation space) presentation-space handle was specified.

PMERR_INV_FORMS_CODE An invalid forms code parameter was specified with DevQueryHardcopyCaps.

PMERR_INV_LENGTH_OR_COUNT An invalid length or count parameter was specified.

DevQueryHardcopyCaps – Query Hardcopy Caps

Related Functions

Prerequisite Functions

- DevOpenDC

Other Related Functions

- DevQueryDeviceNames
- DevQueryCaps

Example Code

The height and width of the capability of the output device is queried for each form code available. Note that a valid device context handle must be passed. This example assumes a DevOpenDC call has been made to obtain the device context handle of a printer.

```
#define INCL_DEV
#include <OS2.H>

HDC hdc;
LONG lStartForm;      /* Form code number from which the query */
                      /* is to start */
LONG lForms;          /* number of forms to query */
/* array of structures containing return information. */
HCINFO ahciHcInfo[5];
LONG lreturn;
int i;
HCINFO height[5];
HCINFO width[5];

lStartForm = 0L;
lForms = 0L;          /* the actual number of forms codes is */
                      /* returned. There will be lreturn */
                      /* copies of the HINFO structure. */

lreturn = DevQueryHardcopyCaps(hdc,
                                lStartForm,
                                lForms,
                                ahciHcInfo);

if (lreturn > 5)
{
    lreturn = 5L;      /* we only want the first five form codes */
}                    /* if there are more than five */

for(i = 0; i < lreturn; i++)
{
    width[lreturn].cx = ahciHcInfo[lreturn].cx;
    height[lreturn].cy = ahciHcInfo[lreturn].cy;
}
```


Chapter 3. Direct Manipulation Functions

This section describes functions that an application would use to initiate or participate in a direct manipulation operation. The following table shows all the direct manipulation (Drg) functions in alphabetic order.

C Name	C Name
DrgAcceptDroppedFiles	DrgQueryNativeRMF
DrgAccessDraginfo	DrgQueryNativeRMFLen
DrgAddStrHandle	DrgQueryStrName
DrgAllocDraginfo	DrgQueryStrNameLen
DrgAllocDragtransfer	DrgQueryTrueType
DrgDeleteDraginfoStrHandles	DrgQueryTrueTypeLen
DrgDeleteStrHandle	DrgReleasePS
DrgDrag	DrgSendTransferMsg
DrgDragFiles	DrgSetDragImage
DrgFreeDraginfo	DrgSetDragitem
DrgFreeDragtransfer	DrgSetDragPointer
DrgGetPS	DrgVerifyNativeRMF
DrgPostTransferMsg	DrgVerifyRMF
DrgPushDraginfo	DrgVerifyTrueType
DrgQueryDragitem	DrgVerifyType
DrgQueryDragitemCount	DrgVerifyTypeSet
DrgQueryDragitemPtr	

DrgAcceptDroppedFiles – Direct Manipulation for Files

```
#define INCL_WINSTDDRAG
```

BOOL DrgAcceptDroppedFiles (HWND Hwnd, PSZ pszPath, PSZ pszTypes,
ULONG ulDefaultOp, ULONG ulReserved)

This function handles the file direct manipulation protocol for a given window.

Parameters

Hwnd (HWND) – input
Window handle.

Handle of calling window.

pszPath (PSZ) – input
Directory.

Directory in which to place the dropped files. If NULL, the files are placed in the current directory.

pszTypes (PSZ) – input
List of types.

A list of types that are acceptable to the drop. This string is of the form: TYPE[,TYPE...].

When this pointer is NULL, any type of file will be accepted.

ulDefaultOp (ULONG) – input
Default drag operation.

Default drag operation for this window. The operation is either DO_MOVE or DO_COPY.

ulReserved (ULONG) – input
Reserved.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Remarks

This function handles the file direct manipulation protocol for a given window. The window responds (DOR_DROP, *usDefaultOp*) to DM_DRAGOVER messages for items with a type matching the acceptable type string and with a rendering mechanism and format of <DRM_OS2FILE,DRF_UNKNOWN>. Not all dragged objects must match this criteria for the drop to be acceptable.

After the drop occurs, this function handles the conversation required to complete the direct manipulation operation for all acceptable objects. A DM_ENDCONVERSATION (DMFL_TARGETFAIL) message is sent to the source when an object is unacceptable.

When an error occurs during a move or copy, the caller is sent a DM_DRAGERROR message. The caller can take corrective action.

As the move or copy operation is successfully completed for each file, a DM_DRAGFILECOMPLETE message is sent to the caller. No message is sent when the operation fails.

The function returns TRUE if the operation is successful and FALSE if an error occurs.

DrgAcceptDroppedFiles – Direct Manipulation for Files

Related Functions

- DrgDragFiles

Example Code

This example uses the DrgAcceptDroppedFiles function to define the direct manipulation protocol of the given window, accept all file types, and use the current directory as the drop directory.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#include <os2.h>

BOOL fSuccess;          /* Indicate success or failure */
HWND Hwnd;              /* Handle of calling window */
PSZ pszPath;            /* Directory in which to place the
                        /* dropped files
PSZ pszTypes;           /* A list of types that are acceptable
ULONG ulDefaultOp;      /* Default drag operation

pszPath = NULL;         /* Drop file in current directory
pszTypes = NULL;        /* Accept any file type
ulDefaultOp = DO_MOVE;  /* Default drag operation is move

fSuccess = DrgAcceptDroppedFiles(Hwnd, pszPath, pszTypes,
                                ulDefaultOp, 0);
```


DrgAccessDraginfo — Access Drag Information

```
#define INCL_WINSTDDRAG
```

BOOL DrgAccessDraginfo (PDRAGINFO pDraginfo)

This function accesses a DRAGINFO structure.

Parameters

pDraginfo (PDRAGINFO) — input

Pointer.

Pointer to the DRAGINFO structure.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_ACCESS_DENIED

The memory block was not allocated properly.

Remarks

This function is used by the target of a drag operation to access a DRAGINFO structure. The address of the structure is passed in a drag message (DM_DRAGOVER, DM_DROP, or DM_DROPHELP).

To release the structure, use the DrgFreeDraginfo function.

Related Functions

- DrgAllocDraginfo
- DrgDrag
- DrgFreeDraginfo
- DrgPushDraginfo

Example Code

This example uses the DrgAccessDraginfo function to make an existing drag information structure (created by the DrgAllocDraginfo function) available.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#include <os2.h>

BOOL      fSuccess;      /* Indicate success or failure */
PDRAGINFO Draginfo;      /* Drag-information structure */

fSuccess = DrgAccessDraginfo(&Draginfo);
```

DrgAddStrHandle – Create String Handle

```
#define INCL_WINSTDDRAG
```

HSTR DrgAddStrHandle (PSZ pszString)

This function creates a handle to a string.

Parameters

pszString (PSZ) – input

String.

String for which a handle is to be created.

Returns

String handle.

NULLHANDLE Error occurred.

Other String handle created.

Possible returns from WinGetLastError

PMERR_INVALID_PARAMETERS

An application parameter value is invalid for its converted PM type. For example: a 4-byte value outside the range –32,768 to +32,767 cannot be converted to a SHORT, and a negative number cannot be converted to a ULONG or USHORT.

PMERR_RESOURCE_DEPLETION

An internal resource depletion error has occurred.

Remarks

The handle can be used by any application to reference the input string.

This function must be called by the source of a drag whenever a string is to be passed in a DRAGINFO structure.

Related Functions

- DrgDeleteStrHandle
- DrgQueryStrName

DrgAddStrHandle — Create String Handle

Example Code

This example calls the DrgAddStrHandle function to create handles for strings that are used in a DRAGITEM structure.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#include <os2.h>

USHORT  ID_ITEM = 1; /* Drag item identifier */
HWND    hwnd;      /* Window handle */
DRAGITEM ditem;     /* DRAGITEM structure */

/* Initialize the DRAGITEM structure */
ditem.hwndItem = hwnd; /* Conversation partner */
ditem.ulItemID = ID_ITEM; /* Identifies item being dragged */
ditem.hstrType = DrgAddStrHandle("DRT_TEXT"); /* Item is text */
ditem.hstrRMF = DrgAddStrHandle("<DRM_OS2FILE,DRF_TEXT>");
ditem.hstrContainerName = DrgAddStrHandle("C:\\");
ditem.hstrSourceName = DrgAddStrHandle("C:\\\\CONFIG.SYS");
ditem.hstrTargetName = DrgAddStrHandle("C:\\\\OS2\\\\CONFIG.SYS");
ditem.cxOffset = 0; /* X-offset of the origin of the */
/* image from the pointer hotspot*/
ditem.cyOffset = 0; /* Y-offset of the origin of the */
/* image from the pointer hotspot*/
ditem.fsControl = 0; /* Source item control flags */
/* object is open */
ditem.fsSupportedOps = 0;
```

DrgAllocDraginfo – Allocate DRAGINFO Structure

```
#define INCL_WINSTDDRAG
```

```
PDRAGINFO DrgAllocDraginfo (ULONG cDitem)
```

This function allocates a DRAGINFO structure.

Parameters

cDitem (ULONG) – input

Number of objects.

Number of objects being dragged. This number must be greater than 0.

Returns

Pointer.

Pointer to the DRAGINFO structure.

NULL Error occurred.

Other The DRAGINFO structure.

Possible returns from WinGetLastError

PMERR_INSUFFICIENT_MEMORY

The operation terminated through insufficient memory.

PMERR_INVALID_PARAMETERS

An application parameter value is invalid for its converted PM type. For example: a 4-byte value outside the range –32,768 to +32,767 cannot be converted to a SHORT, and a negative number cannot be converted to a ULONG or USHORT.

Remarks

This function must be called before the DrgDrag function is called.

The caller can define a default operation for the objects represented by the DRAGINFO structure by modifying the *usOperation* field. If the *usOperation* field is modified, the new value will be sent to the target as the operation whenever a DO_DEFAULT operation would normally be sent. The caller should not modify any other part of the DRAGINFO structure. The DRAGITEM structures associated with the DRAGINFO structure should only be altered with DrgSetDragitem or by using a pointer obtained with DrgQueryDragitemPtr.

Related Functions

- DrgAccessDraginfo
- DrgDrag
- DrgFreeDraginfo
- DrgPushDraginfo

DrgAllocDraginfo — Allocate DRAGINFO Structure

Example Code

This example calls the DrgAllocDraginfo function to create a Drag structure for a single object and uses the new structure to set the DRAGITEM (DrgSetDragitem) structure.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#include <os2.h>

PDRAGINFO  pdinfo;      /* Pointer to DRAGINFO structure */
HWND       hwnd;        /* Handle of calling (source) window */
BOOL       flResult;     /* Result indicator */
DRAGITEM    ditem;      /* DRAGITEM structure */

pdinfo = DrgAllocDraginfo(1); /* Create the DRAGINFO structure */
                        /* Set the drag item */
flResult= DrgSetDragitem(pdinfo, &ditem, (ULONG)sizeof(ditem), 0);
```

DrgAllocDragtransfer – Allocate DRAGTRANSFER Structures

```
#define INCL_WINSTDDRAG
```

```
PDRAGTRANSFER DrgAllocDragtransfer (ULONG cdxfer)
```

This function allocates a specified number of DRAGTRANSFER structures from a single segment.

Parameters

cdxfer (ULONG) – input
Number of structures.

Number of DRAGTRANSFER structures to be allocated. This number must be greater than 0.

Returns

Pointer.

Pointer to an array of DRAGTRANSFER structures.

NULL Error occurred.

Other The array of DRAGTRANSFER structures.

Possible returns from WinGetLastError

PMERR_MEMORY_ALLOCATION_ERR An error occurred during memory management.

PMERR_INSUFFICIENT_MEMORY The operation terminated through insufficient memory.

PMERR_PARAMETER_OUT_OF_RANGE The value of a parameter was not within the defined valid range for that parameter.

Remarks

This function must be called before sending a DM_RENDER message.

Related Functions

- DrgFreeDragtransfer
- DrgSendTransferMsg

Example Code

This example calls the DrgAllocDragtransfer function to allocate a single DRAGTRANSFER structure and adds a pointer to a DRAGITEM structure for an object that will be transferred.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions*/  
#include <os2.h>  
  
PDRAGTRANSFER pResult; /* Pointer to DRAGTRANSFER structure */  
PDRAGITEM pDragitem; /* Pointer to DRAGITEM structure */  
  
pResult = DrgAllocDragtransfer(1);  
  
if (pResult != NULL) /* Indicate DRAGITEM to be transferred */  
    pResult->pditem = pDragitem;
```

DrgDeleteDraginfoStrHandles – Delete DRAGINFO String Handles

```
#define INCL_WINSTDDRAG
```

BOOL DrgDeleteDraginfoStrHandles (PDRAGINFO pDragInfo)

This function deletes each unique string handle in a DRAGINFO structure.

Parameters

pDragInfo (PDRAGINFO) – input
Pointer.

Pointer to the DRAGINFO structure that contains string handles to delete.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INVALID_PARAMETERS

An application parameter value is invalid for its converted PM type. For example: a 4-byte value outside the range –32,768 to +32,767 cannot be converted to a SHORT, and a negative number cannot be converted to a ULONG or USHORT.

Remarks

Using this function is equivalent to calling the DrgDeleteStrHandle function for each unique string in a DRAGINFO structure.

This function must be called by the target of a direct manipulation operation either:

- After processing a DM_DROPHELP message
- or
- After completing the direct manipulation operation begun as a result of a DM_DROP message.

Related Functions

- DrgDeleteStrHandle

Example Code

This example calls the DrgDeleteDraginfoStrHandles function to delete all unique string handles associated with the specified DRAGINFO structure (previously allocated by the DrgAllocDraginfo function).

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */  
#include <os2.h>
```

```
BOOL      fSuccess;      /* Indicate success or failure */  
DRAGINFO  Draginfo;      /* DRAGINFO structure containing string */  
                        /* handles to delete */
```

```
fSuccess = DrgDeleteDraginfoStrHandles (&Draginfo);
```

DrgDeleteStrHandle – Delete String Handle

```
#define INCL_WINSTDDRAG
```

BOOL DrgDeleteStrHandle (HSTR Hstr)

This function deletes a string handle.

Parameters

Hstr (HSTR) – input
String handle.

The string handle to delete.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INVALID_PARAMETERS

An application parameter value is invalid for its converted PM type. For example: a 4-byte value outside the range –32,768 to +32,767 cannot be converted to a SHORT, and a negative number cannot be converted to a ULONG or USHORT.

Remarks

This function must be used to delete a string handle created by the DrgAddStrHandle function.

Related Functions

- DrgAddStrHandle
- DrgDeleteDraginfoStrHandles

Example Code

This example calls the DrgDeleteStrHandle function to delete an existing string handle (returned by a previous call to the DrgAddStrHandle function).

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */  
#include <os2.h>
```

```
BOOL fSuccess;          /* Indicate success or failure */  
HSTR Hstr;              /* String handle */
```

```
fSuccess = DrgDeleteStrHandle (Hstr);
```


DrgDrag —

Drag

```
#define INCL_WINSTDDRAG
```

HWND DrgDrag (HWND hwndSource, PDRAGINFO pDragInfo, PDRAGIMAGE pdimg, ULONG cdlmg, LONG vkTerminate, PVOID pReserved)

This function performs a drag operation.

Parameters

hwndSource (HWND) — input
Window handle.

Handle of the window calling DrgDrag. This window is the source of the drag.

pDragInfo (PDRAGINFO) — input/output
Pointer.

Pointer to the DRAGINFO structure.

pdimg (PDRAGIMAGE) — input
Pointer.

Pointer to an array of DRAGIMAGE structures. These structures describe the images that are to be drawn under the pointing device pointer during the drag.

cdlmg (ULONG) — input
Array size.

Size of the *pdimg* array.

vkTerminate (LONG) — input
Pointing device button.

Pointing device button that ends the drag operation.

VK_BUTTON1 Release of button 1 ends the drag.

VK_BUTTON2 Release of button 2 ends the drag.

VK_BUTTON3 Release of button 3 ends the drag.

VK_ENDDRAG Release of the system-defined direct manipulation button ends the drag. This is the recommended value if the DrgDrag function call is invoked in response to a WM_BEGINDRAG message.

pReserved (PVOID) — input
Reserved.

Must be set to NULL by the caller.

Returns

Window handle.

Handle of window on which the dragged objects were dropped.

NULL Error occurred.

Other Window handle.

Possible returns from WinGetLastError

PMERR_INVALID_HWND An invalid window handle was specified.

PMERR_INVALID_PARAMETERS

An application parameter value is invalid for its converted PM type. For example: a 4-byte value outside the range –32,768 to +32,767 cannot be converted to a SHORT, and a negative number cannot be converted to a ULONG or USHORT.

PMERR_INSUFFICIENT_MEMORY

The operation terminated through insufficient memory.

Remarks

This function:

- Initiates a direct manipulation operation
- Uses the input image to provide visual feedback to the user
- Notifies other windows as the dragged object passes over
- Notifies the destination if the object is dropped.

DrgDrag is called when the system-defined direct-manipulation button is pressed while the pointer is over a window and a pointing device movement follows. As the pointer moves over a potential target, a DM_DRAGOVER message is sent to the target. When the pointer moves from one target window to another, a DM_DRAGLEAVE message is sent to the former target.

If the pointer is over a valid target when the direct-manipulation button is released, a DM_DROP message is sent to the target.

Before the DM_DROP message is sent, the *cxOffset* and *cyOffset* fields are copied from the DRAGIMAGE structures to the corresponding fields in the DRAGITEM structures. The values from the first DRAGIMAGE are copied to the first DRAGITEM, from the second DRAGIMAGE to the second DRAGITEM, and so on. The target can use this information to place the images in the same spatial relationship after the drop. If there are more DRAGITEM structures than there are DRAGIMAGE structures, the *cxOffset* and *cyOffset* from the final DRAGIMAGE are placed in each of the remaining DRAGITEM structures.

The caller can define a default operation for the objects represented by the DRAGINFO structure by modifying the *usOperation* field. If the *usOperation* field is modified, the new value will be sent to the target as the operation whenever a DO_DEFAULT operation would normally be sent. The caller should not modify any other part of the DRAGINFO structure. The DRAGITEM structures associated with the DRAGINFO structure should only be altered with DrgSetDragitem or by using a pointer obtained with DrgQueryDragitemPtr.

The following keys are active during the drag operation:

- | | |
|------------|---|
| Esc | The drag operation is canceled. |
| F1 | A DM_DROPHELP message is posted to the target so that it can provide context help for the drag operation. The drag operation is canceled. |

Before invoking DrgDrag, the caller is responsible for:

- Obtaining a DRAGINFO structure using DrgAllocDraginfo
- Initializing the DRAGITEM structures using DrgSetDragitem.

On return from DrgDrag, the caller must free the structure using DrgFreeDraginfo.

If the dragged objects are not dropped, NULL is returned.

DrgDrag — Drag

Related Functions

Prerequisite Functions

- DrgAllocDraginfo

Other Related Functions

- DrgFreeDraginfo
- DrgSetDragitem

Example Code

This example uses the DrgDrag function to drag a single object in response to the direct-manipulation button being pressed while the pointer is over a drag object. The example shows the initialization of the DRAGITEM, DRAGINFO, and DRAGIMAGE structures used by the DrgDrag function.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#define INCL_WININPUT  /* Window Input Functions */
#include <os2.h>

PDRAGINFO pdinfo; /* Pointer to DRAGINFO structure */
HWND hwnd; /* Handle of calling (source) window */
BOOL flResult; /* Result indicator */
DRAGITEM ditem; /* DRAGITEM structure */
DRAGIMAGE dimg; /* DRAGIMAGE structure */
HBITMAP hbm; /* Bit-map handle */
HWND hwndDrop; /* Handle of drop (target) window */

case WM_BEGINDRAG:

    /******
    /* Initialize the DRAGITEM structure */
    /******
    ditem.hwndItem = hwnd; /* Conversation partner */
    ditem.ulItemID = ID_ITEM; /* Identifies item being dragged*/
    ditem.hstrType = DrgAddStrHandle("DRT_TEXT"); /* Text item */
    ditem.hstrRMF = DrgAddStrHandle("<DRM_OS2FILE,DRF_TEXT>");
    ditem.hstrContainerName = DrgAddStrHandle("C:\\");
    ditem.hstrSourceName = DrgAddStrHandle("C:\\\\CONFIG.SYS");
    ditem.hstrTargetName = DrgAddStrHandle("C:\\OS2\\CONFIG.SYS");
    ditem.cxOffset = 0; /* X-offset of the origin of */
    /* the image from the pointer */
    /* hotspot */
    ditem.cyOffset = 0; /* Y-offset of the origin of */
    /* the image from the pointer */
    /* hotspot */
    ditem.fsControl = 0; /* Source item control flags */
    /* object is open */
    ditem.fsSupportedOps = 0;

    /******
    /* Create the DRAGINFO structure */
    /******
    pdinfo = DrgAllocDraginfo(1);
    if (!pdinfo) return (FALSE); /* If allocation fails, */
    /* return FALSE */

    /******
    /* Initialize the DRAGIMAGE structure */
    /******
    dimg.cb = sizeof(DRAGIMAGE); /* Size control block */
    dimg.cptl = 0;
    dimg.hImage = hbm; /* Image handle passed to */
```

DrgDrag — Drag

```
/* DrgDrag */
dimg.sizlStretch.cx = 20L; /* Size to stretch ico or bmp to*/
dimg.sizlStretch.cy = 20L;
dimg.fl = DRG_BITMAP | /* Flags passed to DrgDrag */
        DRG_STRETCH; /* Stretch to size specified */
                        /* in sizlStretch */
dimg.cxOffset = 0; /* Offset of the origin of */
dimg.cyOffset = 0; /* the image from the pointer */
                        /* hotspot */

/*****
/* Set the drag item */
*****/
flResult= DrgSetDragitem(pdinfo, &ditem, (ULONG)sizeof(ditem),
                        0);

/*****
/* Perform the drag operation: */
/* - Give the user a visual cue by changing the pointer to a */
/* bit map */
/* - Send DM_DRAGOVER messages to the target window (in this */
/* case it is also the source) */
*****/
hwnDDrop = DrgDrag(hwnd, /* Source of the drag */
                  pdinfo, /* Pointer to DRAGINFO structure */
                  (PDRAGIMAGE)&dimg, /* Drag image */
                  1, /* Size of the pdimg array */
                  VK_ENDDRAG, /* Release of direct-manipulation */
                        /* button ends the drag */
                  NULL); /* Reserved */
```

DrgDragFiles – Begin Dragging Files

```
#define INCL_WINSTDDRAG
```

```
BOOL DrgDragFiles (HWND Hwnd, PAPSZ pFiles, PAPSZ pTypes, PAPSZ pTargets,  
                  ULONG cFiles, HPOINTER hptrDrag, ULONG vkTerminate,  
                  BOOL fSourceRender, ULONG ulReserved)
```

This function begins a direct manipulation operation for one or more files.

Parameters

Hwnd (HWND) – input
Window handle.

Handle of calling window.

pFiles (PAPSZ) – input
File names.

The names of the files to be dragged.

pTypes (PAPSZ) – input
File types.

The file types of the files to be dragged.

pTargets (PAPSZ) – input
Target file names.

cFiles (ULONG) – input
Number of files.

Number of files to be dragged.

hptrDrag (HPOINTER) – input
Icon.

Icon to display during the drag.

vkTerminate (ULONG) – input
Button.

Button that ends the drag.

fSourceRender (BOOL) – input
Flag.

Flag to indicate whether the source must perform the move or copy.

TRUE The caller will receive a DM_RENDERFILE message for each file.

FALSE All file manipulation is performed by DrgDragFiles.

ulReserved (ULONG) – input
Reserved.

Returns

Success indicator.

TRUE The drag operation was initiated successfully.

FALSE An error occurred.

DrgDragFiles – Begin Dragging Files

Remarks

This function begins a direct manipulation operation for one or more files. DRAGINFO and DRAGITEM structures are allocated and initialized, and are then used as input to DrgDrag. All of the post-drag conversation required to complete the direct manipulation operation is handled by an object window created by this function.

The caller should set *fSourceRender* to TRUE if it must perform the file manipulation for any of these files. When *fSourceRender* is TRUE, the caller receives a DM_RENDERFILE message as the drag-object window receives a DM_RENDER message. The caller should move or copy the file after receiving the DM_RENDERFILE message. The caller should then send a DM_FILERENDERED message to the drag-object window, and the drag-object window should send a DM_RENDERCOMPLETE message to the target.

When *pTypes* is NULL, the .TYPE EA is interrogated to determine the type for each file in *pFiles*. When *pTypes* is not NULL, the size of the array is expected to be the same as the size of *pFiles*. When any individual pointer in the array is NULL, the .TYPE EA for the corresponding file is read. When .TYPE EA does not exist for any file for which it is needed, a type of DRT_UNKNOWN is used.

When *pTargets* is NULL, the target name for a file will be the same as the source file name with the path information removed. If *pTargets* is not NULL, the size of the array is expected to be the same as the size of *pFiles*. If any individual pointer in the array is NULL, the target name for the corresponding file will match the source name minus the path information.

The rendering mechanism and format for each file is: <DRM_OS2FILE,DRF_UNKNOWN>.

When an error occurs during the move or copy, the caller is sent a DM_DRAGERROR message. The caller can take corrective action.

As the operation is complete for each file in the list, a DM_DRAGFILECOMPLETE message is sent to the caller of DrgDragFiles. The caller is thus notified that resources can be freed for a particular file.

This function returns TRUE if the drag operation was initiated successfully and FALSE if an error occurred.

Related Functions

- DrgAcceptDroppedFiles

DrgDragFiles – Begin Dragging Files

Example Code

This example calls the DrgDragFiles function to begin direct manipulation for a single file object, using the same source and target name, and determining the file type based on the file's type EA.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#define INCL_WININPUT  /* Window Input Functions */
#include <os2.h>

BOOL    fSuccess;      /* Indicate success or failure */
HWND    Hwnd;          /* Handle of calling window */
PSZ     pFiles[1];     /* The names of the files to be dragged */
PSZ     pTypes[1];     /* The file types of the files to be
                        /* dragged
PSZ     pTargets[1];   /* The target file names
HPOINTER hpPtrDrag;    /* Icon to display during drag

pFiles[0] = "FILENAME.EXT"; /* Copy file name to string array */
pTargets[0] = NULL;        /* Use source name as target name */
pTypes[0] = NULL;         /* Query type EA to determine file type */

fSuccess = DrgDragFiles(Hwnd, pFiles, pTypes, pTargets, 1,
                        hpPtrDrag, VK_BUTTON2, FALSE, 0L);
```

DrgFreeDraginfo – Free DRAGINFO Structure

```
#define INCL_WINSTDDRAG
```

```
BOOL DrgFreeDraginfo (PDRAGINFO pDraginfo)
```

This function frees a DRAGINFO structure allocated by DrgAllocDraginfo.

Parameters

pDraginfo (PDRAGINFO) – input
Pointer.

Pointer to the DRAGINFO structure.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_MEMORY_DEALLOCATION_ERR An error occurred during memory management.

PMERR_SOURCE_SAME_AS_TARGET The direct manipulation source and target process are the same.

Remarks

DrgFreeDraginfo fails with an error of PMERR_SOURCE_SAME_AS_TARGET if it is called by the process that called DrgDrag before DrgDrag returns. When a process is performing a drag operation between two of its own windows, this prevents the source window from freeing the DRAGINFO structure before the target window finishes processing.

Related Functions

Prerequisite Functions

- DrgAllocDraginfo

Other Related Functions

- DrgDrag
- DrgAccessDraginfo
- DrgPushDraginfo

DrgFreeDraginfo — Free DRAGINFO Structure

Example Code

This example calls the DrgFreeDraginfo function to free an existing DRAGINFO structure allocated by the DrgAllocDraginfo function after a drag operation has completed.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#include <os2.h>
```

```
BOOL      fSuccess;    /* Indicate success or failure */
PDRAGINFO pdinfo;      /* Pointer to DRAGINFO structure */
HWND      hwnd;        /* Handle of calling (source) window */
DRAGIMAGE dimg;        /* DRAGIMAGE structure */
HWND      hwndDrop;     /* Handle of drop (target) window */
```

```
/******
/* Perform the drag operation:
/* - Give the user a visual cue by changing the pointer to a
/*   bit map
/* - Send DM_DRAGOVER messages to the target window (in this
/*   case it is also the source)
/******
hwndDrop = DrgDrag(hwnd,      /* Source of the drag */
                  pdinfo,     /* Pointer to DRAGINFO structure */
                  (PDRAGIMAGE)&dimg, /* Drag image */
                  1,          /* Size of the pdimg array */
                  VK_ENDDRAG,  /* Release of drag button */
                  /* Terminates the drag */
                  NULL);       /* Reserved */
```

```
fSuccess = DrgFreeDraginfo(&pdinfo);
```

DrgFreeDragtransfer – Free DRAGTRANSFER Storage

```
#define INCL_WINSTDDRAG
```

BOOL DrgFreeDragtransfer (PDAGTRANSFER pdxfer)

This function frees the storage associated with a DRAGTRANSFER structure.

Parameters

pdxfer (PDAGTRANSFER) – input
Pointer.

Pointer to the DRAGTRANSFER structures to be freed.

Returns

Return code.

0 The structure was freed.

Other Deallocation failed.

Possible returns from WinGetLastError

PMERR_MEMORY_DEALLOCATION_ERR An error occurred during memory management.

Remarks

This function frees the DRAGTRANSFER structures allocated by calls to DrgAllocDragtransfer. When all of the DRAGTRANSFER structures have been freed, the memory block containing the DRAGTRANSFER array is deallocated.

Related Functions

- DrgAllocDragtransfer

Example Code

This example calls the DrgFreeDragtransfer function to free an existing DRAGTRANSFER structure allocated by the DrgAllocDragtransfer function.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */  
#include <os2.h>
```

```
BOOL            fSuccess; /* Indicate success or failure        */  
PDAGTRANSFER dxfer;     /* Pointer to DRAGTRANSFER structure */
```

```
fSuccess = DrgFreeDragtransfer(&dxfer);
```

DrgGetPS — Get Drag Presentation Space

```
#define INCL_WINSTDDRAG
```

HPS DrgGetPS (HWND Hwnd)

This function gets a presentation space that is used to provide target feedback to the user during a drag operation.

Parameters

Hwnd (HWND) — input
Window handle.

Handle of the window for which presentation space is required.

Returns

Presentation-space handle.

Presentation-space handle used for drawing in the window.

NULLHANDLE Error occurred.

Possible returns from WinGetLastError

PMERR_INVALID_HWND

An invalid window handle was specified.

PMERR_NOT_DRAGGING

A drag operation is not in progress at this time.

Remarks

This function returns a handle to a presentation space that can be used for drawing while a direct manipulation operation is in progress.

DrgGetPS is called only during a direct manipulation operation. This function is called only after a DM_DRAGOVER, DM_DRAGLEAVE, or DM_DROP message has been received.

In order to draw target emphasis, an application must use DrgGetPS and DrgReleasePS to unlock its window.

The presentation space created with DrgGetPS must be freed with DrgReleasePS.

Related Functions

- DrgReleasePS

DrgGetPS – Get Drag Presentation Space

Example Code

This example uses the DrgGetPS function to get a presentation space handle which is used during drag operations such as loading a drag bit map. When finished with the presentation space, release it with the DrgReleasePS function.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#include <os2.h>

HPS  hps;          /* Presentation space handle */
HWND hwnd;         /* Handle of the window for which
                    /* presentation space is required */

case DM_DRAGOVER:
    hps = DrgGetPS(hwnd);

DrawTargetEmphasis(hps, hwnd);
DrgReleasePS(hps);
```

DrgPostTransferMsg – Post Drag Message

```
#define INCL_WINSTDDRAG
```

BOOL DrgPostTransferMsg (HWND *hwndTo*, ULONG *uMsgId*, PDRA^GTRANSFER *pdxfer*,
ULONG *fs*, ULONG *uiReserved*, BOOL *fRetry*)

This function posts a message to the other application involved in the direct manipulation operation.

Parameters

hwndTo (HWND) – input
Window handle.

Window handle to which the message is to be posted:

Target *hwndItem* in the DRAGITEM structure.

Source *hwndClient* in the DRAGTRANSFER structure.

uMsgId (ULONG) – input
Message identifier.

Identifier of the message to be posted. DM_RENDERCOMPLETE is the only valid message.

pdxfer (PDRA^GTRANSFER) – input
Pointer.

Pointer to the DRAGTRANSFER structure.

fs (ULONG) – input
Flags.

The flags to be passed in the *param2* parameter of the message.

uiReserved (ULONG) – input
Reserved.

This must be 0.

fRetry (BOOL) – input
Retry indicator.

TRUE If the destination queue is full, the message posting is retried at 1-second intervals until the message is posted successfully.

In this case, DrgPostTransferMsg dispatches any messages in the queue by calling WinPeekMsg and WinDispatchMsg in a loop. The application can receive messages sent by other applications while it is trying to post drag transfer messages.

FALSE The call returns FALSE without retrying.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

DrgPostTransferMsg – Post Drag Message

Remarks

The *usReply* field in the DRAGTRANSFER structure is set to 0 before the message is posted. If the posting fails for any reason, FALSE is returned.

Related Functions

- DrgSendTransferMsg

Example Code

This example calls the DrgPostTransferMsg function to respond to a DM_RENDER message from the target. The response consists of a DM_RENDERCOMPLETE message, plus a flag indicating whether the render was successful (DMFL_RENDEROK) or not (DMFL_RENDERFAIL).

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#include <os2.h>

MPARAM      mp1;      /* Message parameter 1 */
BOOL        fSuccess; /* Indicate success or failure */
BOOL        Rendered; /* Success of render operation */
PDRAGTRANSFER pdxfer; /* Pointer to DRAGTRANSFER structure */

case DM_RENDER:
    pdxfer = (PDRAGTRANSFER)PVOIDFROMMP(mp1); /* Get DRAGTRANSFER */
                                              /* structure */

    /* Attempt to render file */

    if (Rendered)
    {
        fSuccess = DrgPostTransferMsg(pdxfer->pditem,
                                      DM_RENDERCOMPLETE,
                                      pdxfer,
                                      DMFL_RENDEROK,
                                      0,FALSE);

        return (MRESULT)TRUE;
    }
    else
    {
        fSuccess = DrgPostTransferMsg(pdxfer->pditem,
                                      DM_RENDERCOMPLETE,
                                      pdxfer,
                                      DMFL_RENDERFAIL,
                                      0,FALSE);

        return (MRESULT)FALSE;
    }
}
```

DrgPushDraginfo – Access a DRAGINFO Structure

```
#define INCL_WINSTDDRAG
```

BOOL DrgPushDraginfo (PDRAGINFO pDraginfo, HWND hwndDest)

This function gives a process access to a DRAGINFO structure.

Parameters

pDraginfo (PDRAGINFO) – input
Pointer.

Pointer to the DRAGINFO structure.

hwndDest (HWND) – input
Window handle.

Handle of the window whose process is to be given access to a DRAGINFO structure.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from GetLastError

PMERR_ACCESS_DENIED The memory block was not allocated properly.

PMERR_INSUFFICIENT_MEMORY The operation terminated through insufficient memory.

Remarks

The receiving process is responsible for:

1. Deleting the string handles in the DRAGINFO structure with DrgDeleteDraginfoStrHandles
2. Freeing the DRAGINFO structure using DrgFreeDraginfo.

Related Functions

- DrgAllocDraginfo
- DrgDrag
- DrgAccessDraginfo
- DrgFreeDraginfo

DrgPushDraginfo — Access a DRAGINFO Structure

Example Code

This example calls the DrgPushDraginfo function to grant access to a DRAGINFO structure to the process owning the specified window handle. The DRAGINFO structure was previously allocated using the DrgAllocDraginfo function.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */  
#include <os2.h>
```

```
BOOL      fSuccess;      /* Indicate success or failure      */  
DRAGINFO  Draginfo;      /* Pointer to DRAGINFO structure    */  
HWND      hwndDest;      /* Handle of window whose process will */  
                        /* will be given access to the DRAGINFO */  
                        /* structure                          */
```

```
fSuccess = DrgPushDraginfo(&Draginfo,hwndDest);
```


DrgQueryDragitem – Get DRAGITEM Structure

```
#define INCL_WINSTDDRAG
```

```
BOOL DrgQueryDragitem (PDRAGINFO pDragInfo, ULONG cbBuffer, PDRAGITEM pDragItem,  
                        ULONG lItem)
```

This function returns a DRAGITEM structure used in the direct manipulation operation.

Parameters

pDragInfo (PDRAGINFO) – input
Pointer.

Pointer to the DRAGINFO structure from which the DRAGITEM structure is obtained.

cbBuffer (ULONG) – input
Number of bytes.

Maximum number of bytes to copy.

pDragitem (PDRAGITEM) – output
Pointer.

Pointer to the buffer into which the DRAGITEM structure is copied.

lItem (ULONG) – input
DRAGITEM index.

Zero-based index of the DRAGITEM to be returned.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Remarks

This function returns the DRAGITEM structure identified by *lItem*.

Related Functions

- DrgSetDragitem
- DrgQueryDragitemPtr

DrgQueryDragitem – Get DRAGITEM Structure

Example Code

This example calls the DrgQueryDragitem function to return the entirety of the first DRAGITEM structure in the given DRAGINFO structure, after which it obtains the source window handle.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#include <os2.h>

BOOL      fSuccess;      /* Indicate success or failure */
DRAGINFO  Draginfo;      /* DRAGINFO structure from which the
                          /* DRAGITEM structure is obtained */
ULONG     cbBuffer;      /* Maximum number of bytes to copy */
DRAGITEM  Dragitem;      /* Buffer into which the DRAGITEM
                          /* structure is copied */
ULONG     iItem;         /* Zero-based index of the DRAGITEM
                          /* to be returned */
HWND      hwndSource;    /* Source window handle for the drag */

cbBuffer = sizeof(DRAGITEM); /* Copy entire DRAGITEM structure */
iItem = 0;                  /* Return first DRAGITEM */

fSuccess = DrgQueryDragitem(&Draginfo,cbBuffer,&Dragitem,iItem);

hwndSource = Dragitem.hwndItem; /* Obtain source window handle */
```

DrgQueryDragitemCount – Get Dragged Object Count

```
#define INCL_WINSTDDRAG
```

ULONG DrgQueryDragitemCount (PDRAGINFO pDragInfo)

This function returns the number of objects being dragged during the current direct manipulation operation.

Parameters

pDragInfo (PDRAGINFO) – input
Pointer.

Pointer to the DRAGINFO structure for which the number of dragged objects is requested.

Returns

Number of objects.

Number of objects being dragged.

Example Code

This example calls the DrgQueryDragitemCount function to return the number of DRAGITEM structures in the corresponding DRAGINFO structure, which maps to the number of objects being dragged.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#include <os2.h>

ULONG      cDitem;      /* Number of objects being dragged */
PDRAGINFO  pDraginfo;   /* DRAGINFO structure queried for the */
                        /* number of drag objects */

cDitem = DrgQueryDragitemCount(pDraginfo);
```

DrgQueryDragitemPtr – Get Pointer to DRAGITEM Structure

```
#define INCL_WINSTDDRAG
```

PDRAGITEM DrgQueryDragitemPtr (PDRAGINFO pDragInfo, ULONG ulIndex)

This function returns a pointer to the DRAGITEM structure used in the direct manipulation operation.

Parameters

pDraginfo (PDRAGINFO) – input
Pointer.

Pointer to the DRAGINFO structure from which the DRAGITEM structure is obtained.

ulIndex (ULONG) – input
DRAGITEM index.

Zero-based index of the DRAGITEM structure for which the pointer is to be returned.

Returns

Pointer.

Pointer to the DRAGITEM structure.

Remarks

This function returns a pointer to *ulItemID* in the DRAGITEM structure used in the direct manipulation operation.

Related Functions

- DrgQueryDragitem

Example Code

This example calls the DrgQueryDragitemPtr function to return a pointer to first DRAGITEM structure in the given DRAGINFO structure, after which it obtains the source window handle.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#include <os2.h>

PDRAGITEM pDragitem; /* DRAGITEM structure pointer */
PDRAGINFO Draginfo; /* DRAGINFO structure from which the */
                /* DRAGITEM structure is obtained */
ULONG      ulIndex; /* Zero-based index of the DRAGITEM */
                /* structure pointer to be returned */
HWND       hwndSource; /* Source window handle for the drag */

USHORT      usn = 0; /* Return pointer to first DRAGITEM */

pDragitem = DrgQueryDragitemPtr(&Draginfo,usn);

hwndSource = pDragitem->hwndItem; /* Obtain source window handle */
```

DrgQueryNativeRMF –

Get Format of a Dragged Object

```
#define INCL_WINSTDDRAG
```

BOOL DrgQueryNativeRMF (PDRAGITEM pDragItem, ULONG cbBuflen, PCHAR ppBuffer)

This function obtains the ordered pair that represents the native rendering mechanism and format of the dragged object.

Parameters

pDragItem (PDRAGITEM) – input
Pointer.

Pointer to the DRAGITEM structure whose native rendering mechanism and format are to be obtained.

cbBuflen (ULONG) – input
Number of bytes.

Maximum number of bytes to copy to the buffer.

ppBuffer (PCHAR) – output
Pointer.

Pointer to the buffer in which the null-terminated string is to be returned.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from GetLastError

PMERR_INVALID_PARAMETERS

An application parameter value is invalid for its converted PM type. For example: a 4-byte value outside the range –32,768 to +32,767 cannot be converted to a SHORT, and a negative number cannot be converted to a ULONG or USHORT.

Remarks

If the rendering mechanism and format string for the object are NULL, FALSE is returned. If TRUE is returned, the format of the string is: <MECHANISM,FORMAT>.

The native rendering mechanism and format are the first ordered pair, or the first ordered pair produced by a cross product, in the string associated with *hstrRMF* in the DRAGITEM structure.

DrgQueryNativeRMFLen can be used to determine the size of the buffer required to hold the string returned by this function.

Related Functions

Prerequisite Functions

- DrgQueryNativeRMFLen

Other Related Functions

- DrgVerifyNativeRMF

DrgQueryNativeRMF — Get Format of a Dragged Object

Example Code

This example shows how to obtain the window handle of the source of a drag item.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#define INCL_DOSMEMMGR /* Memory Management Functions for */
                        /* DosSubAlloc */

#include <OS2.H>

DRAGITEM ditem;
PVOID pMem;
PSZ pszBuffer;
ULONG cb;
BOOL rc, fResult;

cb = DrgQueryNativeRMFLen(&ditem) + 1;

rc = DosSubAlloc(pMem, (PVOID *) pszBuffer, cb);

if (!rc)
{
    fResult = DrgQueryNativeRMF(&ditem, cb, pszBuffer);
}
```

DrgQueryNativeRMFLen – Get String Length for Native RMF of Dragged Object

```
#define INCL_WINSTDDRAG
```

ULONG DrgQueryNativeRMFLen (PDRAGITEM pDragItem)

This function obtains the length of the string representing the native rendering mechanism and format of the dragged object.

Parameters

pDragItem (PDRAGITEM) – input
Pointer.

Pointer to the DRAGITEM structure whose native rendering mechanism and format string length are to be obtained.

Returns

String length.

String length of the ordered pair:

0 Error occurred.

Other String length of the ordered pair, excluding the null-terminating byte.

Possible returns from GetLastError

PMERR_INVALID_PARAMETERS

An application parameter value is invalid for its converted PM type. For example: a 4-byte value outside the range –32,768 to +32,767 cannot be converted to a SHORT, and a negative number cannot be converted to a ULONG or USHORT.

Remarks

This function is used to determine the size of the buffer that contains the string representing the native rendering mechanism and format of the dragged object.

If the input string handle is NULLHANDLE or not valid, a length of 0 is returned.

Related Functions

- DrgQueryNativeRMF

DrgQueryNativeRMFLen — Get String Length for Native RMF of Dragged Object

Example Code

This example shows how to obtain the window handle of the source of a drag item.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#define INCL_DOSMEMMGR /* Memory Management Functions for */
                        /* DosSubAlloc */
#include <OS2.H>

DRAGITEM ditem;
PVOID    pMem;
PSZ      pszBuffer;
ULONG    cb;
BOOL     rc, fResult;

cb = DrgQueryNativeRMFLen(&ditem) + 1;

rc = DosSubAlloc(pMem, (PVOID *) pszBuffer, cb);

if (!rc)
{
    fResult = DrgQueryNativeRMF(&ditem, cb, pszBuffer);
}
```


DrgQueryStrName — Get String Contents

```
#define INCL_WINSTDDRAG
```

ULONG DrgQueryStrName (HSTR Hstr, ULONG cbBuflen, PSZ pszBuffer)

This function gets the contents of a string associated with a string handle.

Parameters

Hstr (HSTR) — input
String handle.

The handle must have been created with DrgAddStrHandle.

cbBuflen (ULONG) — input
Number of bytes.

Maximum number of bytes to copy.

pszBuffer (PSZ) — output
Buffer.

Buffer where the null-terminated string is returned.

Returns

Number of bytes.

Number of bytes written to *pszBuffer*.

Possible returns from WinGetLastError

PMERR_INVALID_PARAMETERS

An application parameter value is invalid for its converted PM type. For example: a 4-byte value outside the range -32,768 to +32,767 cannot be converted to a SHORT, and a negative number cannot be converted to a ULONG or USHORT.

Remarks

This function should be called whenever the contents of a string referenced by a drag string handle are required. If the input string handle is NULLHANDLE or not valid, a null string is returned.

Related Functions

Prerequisite Functions

- DrgQueryStrNameLen

Other Related Functions

- DrgAddStrHandle

DrgQueryStrName — Get String Contents

Example Code

This example shows how to obtain the contents of a string given that the string handle is known. The string handle must have been originally created with the DrgAddStrHandle function.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#define INCL_DOSMEMMGR /* Memory Management Functions for */
                        /* DosAllocMem */
#include <OS2.H>

HSTR hstr; /* Handle to a string. The handle must */
           /* have been created with */
           /* DrgAddStrHandle. */
PSZ pBuffer; /* Buffer where the null-terminated */
             /* string is returned */
ULONG ulStrlen; /* String length */
ULONG ulBytesRead; /* Number of bytes read */
ULONG rc; /* Return code */

ulStrlen = DrgQueryStrNameLen(hstr) + 1;

rc = DosAllocMem((PVOID *) pBuffer,
                (LONG)ulStrlen,
                fPERM |
                PAG_COMMIT);

/*****
/* The ulBytesRead parameter contains the number of bytes
/* actually written to the memory pointed to by pBuffer
*****/
ulBytesRead = DrgQueryStrName(hstr,
                             ulStrlen, /* Number of bytes to copy */
                             pBuffer);
```

DrgQueryStrNameLen — Get String Length

```
#define INCL_WINSTDDRAG
```

ULONG DrgQueryStrNameLen (HSTR Hstr)

This function gets the length of a string associated with a string handle.

Parameters

Hstr (HSTR) — input
String handle.

The handle must be created with DrgAddStrHandle.

Returns

String length.

0 The string handle is NULLHANDLE or is not valid.

Other The length of the string associated with the string handle, excluding the null terminating byte.

Possible returns from WinGetLastError

PMERR_INVALID_PARAMETERS

An application parameter value is invalid for its converted PM type. For example: a 4-byte value outside the range -32,768 to +32,767 cannot be converted to a SHORT, and a negative number cannot be converted to a ULONG or USHORT.

Remarks

This function should be called before calling the DrgQueryStrName function. It is used to determine and allocate the buffer size for the string associated with the string handle. If the input string handle is NULLHANDLE or not valid, a length of 0 is returned.

Related Functions

- DrgQueryStrName

DrgQueryStrNameLen – Get String Length

Example Code

This example shows how to obtain the length of a string given that the string handle is known. The string handle must have been originally created with the DrgAddStrHandle function.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#define INCL_DOSMEMMGR /* Memory Management Functions for      */
                        /* DosAllocMem                          */
#include <OS2.H>

HSTR  hstr;           /* Handle to a string. The handle must */
                        /* have been created with              */
                        /* DrgAddStrHandle.                    */
PSZ   pBuffer;        /* Buffer where the null-terminated     */
                        /* string is returned                  */
ULONG ulStrlen;        /* String length                        */
ULONG ulBytesRead;     /* Number of bytes read                */
ULONG rc;              /* Return code                          */

ulStrlen = DrgQueryStrNameLen(hstr) + 1;

rc = DosAllocMem((PVOID *) pBuffer,
                 (LONG)ulStrlen,
                 fPERM |
                 PAG_COMMIT);

/*****
/* The ulBytesRead parameter contains the number of bytes
/* actually written to the memory pointed to by pBuffer
*****/
ulBytesRead = DrgQueryStrName(hstr,
                              ulStrlen, /* Number of bytes to copy */
                              pBuffer);
```

DrgQueryTrueType – Get True Type of Dragged Object

```
#define INCL_WINSTDDRAG
```

BOOL DrgQueryTrueType (PDRAGITEM pDragitem, ULONG cbBuflen, PSZ pszBuffer)

This function obtains the true type of a dragged object.

Parameters

pDragitem (PDRAGITEM) – input
Pointer.

Pointer to the DRAGITEM structure whose true type is to be obtained.

cbBuflen (ULONG) – input
Number of bytes.

Maximum number of bytes to copy to the buffer.

pszBuffer (PSZ) – output
Buffer.

Buffer in which the null-terminated string is to be returned.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INVALID_PARAMETERS

An application parameter value is invalid for its converted PM type. For example: a 4-byte value outside the range –32,768 to +32,767 cannot be converted to a SHORT, and a negative number cannot be converted to a ULONG or USHORT.

Remarks

The true type of an object is the first type in the string referenced by *hstrType* in the DRAGITEM structure.

This function can be called after calling the DrgQueryTrueTypeLen function. If the type string for the object is NULLHANDLE, FALSE is returned.

Related Functions

Prerequisite Functions

- DrgQueryTrueTypeLen

Other Related Functions

- DrgVerifyTrueType

DrgQueryTrueType – Get True Type of Dragged Object

Example Code

This example shows how to obtain the true type of an object.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#include <OS2.H>

BOOL      fSuccess;      /* Return value */
DRAGITEM Dragitem;       /* DRAGITEM structure whose true type */
                        /* is to be obtained */

char szBuffer[32];       /* Buffer in which the null-terminated */
                        /* string is to be returned */

fSuccess = DrgQueryTrueType(&Dragitem,
                           sizeof(szBuffer),
                           szBuffer);
```

DrgQueryTrueTypeLen – Get String Length for True Type of Dragged Object

```
#define INCL_WINSTDDRAG
```

ULONG DrgQueryTrueTypeLen (PDRAGITEM pDragItem)

This function obtains the length of the string that represents the true type of a dragged object.

Parameters

pDragItem (PDRAGITEM) – input
Pointer.

Pointer to the DRAGITEM structure whose true type length is to be obtained.

Returns

String length.

0 Error occurred.

Other The length of the first element of the character string associated with *hstrType*, excluding the null-terminating byte.

Possible returns from WinGetLastError

PMERR_INVALID_PARAMETERS

An application parameter value is invalid for its converted PM type. For example: a 4-byte value outside the range –32,768 to +32,767 cannot be converted to a SHORT, and a negative number cannot be converted to a ULONG or USHORT.

Remarks

This function can be used to determine the buffer size to allocate for the string representing the true type of a dragged object. The true type of an object is the first type in the type string referenced by *hstrType* in the DRAGITEM structure.

This function can be called before calling the DrgQueryTrueType function.

If the input string handle is NULLHANDLE or not valid, a length of 0 is returned.

Related Functions

- DrgQueryTrueType

DrgQueryTrueTypeLen – Get String Length for True Type of Dragged Object

Example Code

This example shows how to obtain the length of the true type string with the DrgQueryTrueTypeLen function.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#define INCL_DOSMEMMGR /* Memory Management Functions for */
                        /* DosAllocMem */
#include <OS2.H>

PSZ      pszBuffer;      /* Buffer in which the DRAGITEM */
                        /* structure is stored */
BOOL      fSuccess;      /* Return value */
DRAGITEM Dragitem;      /* DRAGITEM structure whose true type */
                        /* length is to be obtained */
ULONG     rc;            /* Return code */
ULONG     ulLength;      /* String length of dragged object */

ulLength = DrgQueryTrueTypeLen(&Dragitem) + 1;

rc = DosAllocMem((PVOID *) pszBuffer, ulLength, fPERM);

fSuccess = DrgQueryTrueType(&Dragitem, ulLength, pszBuffer);
```


DrgReleasePS – Release Presentation Space

```
#define INCL_WINSTDDRAG
```

BOOL DrgReleasePS (HPS Hps)

This function releases a presentation space obtained by using the DrgGetPS function.

Parameters

Hps (HPS) – input

Presentation-space handle.

Handle of the presentation space to release.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_NOT_DRAGGING

A drag operation is not in progress at this time.

Remarks

Only presentation spaces created with DrgGetPS can be released using this function.

The presentation-space handle should not be used after this function.

Related Functions

Prerequisite Functions

- DrgGetPS

Example Code

In this example the presentation space handle is retrieved, a bit map is loaded, and the presentation space is released with the DrgReleasePS function.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#include <os2.h>
#define ID_BITMAP 255
HPS hps;
HWND hwnd;

case DM_DRAGOVER:
    hps = DrgGetPS(hwnd);

    DrawTargetEmphasis(hps, hwnd);
    DrgReleasePS(hps);
```

DrgSendTransferMsg – Send Drag Message

```
#define INCL_WINSTDDRAG
```

**MRESULT DrgSendTransferMsg (HWND hwndTo, ULONG ulMsgid, MPARAM mpParam1,
MPARAM mpParam2)**

This function sends a message to the other application involved in the direct manipulation operation.

Parameters

hwndTo (HWND) – input
Window handle.

Window handle to which the message is to be sent:

Target *hwndItem* in the DRAGITEM structure.

Source *hwndClient* in the DRAGTRANSFER structure.

ulMsgid (ULONG) – input
Message identifier.

Identifier of the message to be sent. Valid messages are:

DM_ENDCONVERSATION
DM_RENDER
DM_RENDERPREPARE

mpParam1 (MPARAM) – input
mp1 for the message.

mpParam2 (MPARAM) – input
mp2 for the message.

Returns

Message-return data.

Remarks

If the message to be sent is DM_RENDER or DM_RENDERCOMPLETE, the *usReply* field in DRAGTRANSFER is set to 0 before the message is sent. If the message cannot be sent, FALSE is returned.

When the message to be sent is DM_RENDER, DosGiveSeg is called. DosGiveSeg gives access to the DRAGTRANSFER structure to the process that owns the window indicated by *hwndTo*. The use count for the segment in which the DRAGTRANSFER structure exists is incremented.

The process to which the message is being sent must call DrgFreeDragtransfer for the DRAGTRANSFER structure before the segment can be freed.

Related Functions

- DrgPostTransferMsg

DrgSendTransferMsg —

Send Drag Message

Example Code

This function is used to send a message from one window to another when a direct manipulation is in progress. In this example, the function is used to inform the target that the operation is complete and successful.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#include <os2.h>
```

```
PDRAGINFO pdinfo;
MPARAM    mpl;
TID       tid;
```

```
case DM_DROP:
    pdinfo = (PDRAGINFO) mpl;
```

```
    /******
    /* If this is a copy operation, spawn a thread to do the copy */
    /******
    if (pdinfo->usOperation == DO_COPY)
    {
        DosCreateThread (&tid, CopyThread, pdinfo, FALSE, 4096);
    }
    break;
```

```
void Copy Thread (PDRAGINFO pdinfo)
{
```

```
    PDRAGITEM pditem;
    USHORT    i;
    ULONG     flResult;
    HAB       hab;
    HMQ       hmq;
    char       szSource[CCH_MAXPATH];
    char       szTarget[CCH_MACPATH];
```

```
    /******
    /* DrgSendTransferMsg needs a message queue, so create one for */
    /* this thread */
    /******
    hab = WinInitialize (0);
    hmq = WinCreateMsgQueue (hab, 0);
```

```
    /******
    /* Try to copy each item that was dragged */
    /******
    for (i = 0; i < pdinfo->cditem; i++)
    {
```

```
        /******
        /* Get a pointer to the DRAGITEM */
        /******
        pditem = DrgQueryDragitemPtr (pdinfo, i);
```

```
        /******
        /* If we could query the source and target names, and the */
        /* copy was successful, return success */
        /******
        if (DrgQueryStrName (pditem->hstrSourceName, sizeof (szSource),
                             szSource)
            DrgQueryStrName (pditem->hstrTargetName, sizeof (szTarget),
                             szTarget)
            !DosCopy (szSource, szTarget, 0))
        {
```

```
            flResult = DMFL_TARGETSUCCESSFUL;
        }
```

DrgSendTransferMsg – Send Drag Message

```

/*****
/* Otherwise, return failure
*****/
else
{
    flResult = DMFL_TARGETFAIL;
}

/*****
/* Let the source know we're done with this item
*****/
DrgSendTransferMsg (pditem->hwndItem, DM_ENDCONVERSATION,
                    (MPARAM) pditem->ulItemID,
                    (MPARAM) flResult);
}

WinDestroyMsgQueue (hmq);
WinTerminate (hab);
}
```

DrgSetDragImage —

Set Drag Image

```
#define INCL_WINSTDDRAG
```

```
BOOL DrgSetDragImage (PDRAGINFO pDraginfo, PDRAGIMAGE pdimg, ULONG cdimg,  
PVOID pReserved)
```

This function sets the image that is being dragged.

Parameters

pDraginfo (PDRAGINFO) — input
Pointer.

Pointer to the DRAGINFO structure representing the drag operation for which the pointer is to be set.

pdimg (PDRAGIMAGE) — input
Pointer.

Pointer to an array of DRAGIMAGE structures. These structures describe the images to be drawn under the pointer during the drag.

cdimg (ULONG) — input
Array size.

Size of the *pdimg* array.

pReserved (PVOID) — input
Reserved.

Must be set to NULL by the caller.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_ACCESS_DENIED

The memory block was not allocated properly.

PMERR_INVALID_PARAMETERS

An application parameter value is invalid for its converted PM type. For example: a 4-byte value outside the range -32,768 to +32,767 cannot be converted to a SHORT, and a negative number cannot be converted to a ULONG or USHORT.

PMERR_INSUFFICIENT_MEMORY

The operation terminated through insufficient memory.

Remarks

The image that is set with DrgSetDragImage is used only while the pointer is over the target that made the call. If the pointer leaves the original target, the new target can specify an image by calling DrgSetDragImage.

If the new target does not call DrgSetDragImage, the original image that was supplied on the call to DrgDrag is used.

Related Functions

- DrgSetDragPointer

Example Code

This example sets the icon image that is displayed during a direct manipulation operation.

```
#define INCL_GPIBITMAPS /* GPI Bit Map Functions */
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#include <os2.h>
#define ID_BITMAP 257 /* .rc file: "bitmap 257 drgimage.bmp" */
HPS hps; /* Presentation space handle */
BOOL flResult;
HAB hab;
PDRAGINFO pdinfo;
DRAGIMAGE dimg;
HBITMAP hbm; /* Bit-map handle */
HWND hwnd;

/*****
/* Load a bit map for use as a drag image
*****/

case WM_CREATE:
    hps = WinGetPS(hwnd);

    hbm = GpiLoadBitmap(hps, 0L, ID_BITMAP, 20L, 20L);
    WinReleasePS(hps);
    break;

case DM_DRAGOVER:

/*****
/* Initialize the DRAGIMAGE structure
*****/

    dimg.cb = sizeof(DRAGIMAGE); /* Size control block */
    dimg.cptl = 0;
    dimg.hImage = hbm; /* Image handle passed to */
    /* DrgDrag */
    dimg.sizlStretch.cx = 20L; /* Size to stretch ico or */
    dimg.sizlStretch.cy = 20L; /* bmp to */
    dimg.fl = DRG_BITMAP |
    DRG_STRETCH; /* Stretch to size specified */
    dimg.cxOffset = 0; /* Offset of the origin of */
    dimg.cyOffset = 0; /* the image from the pointer*/
    /* hotspot */

/*****
/* Set the drag image
*****/

flResult= DrgSetDragImage(pdinfo, &dimg, (ULONG)sizeof(dimg), NULL);
```

DrgSetDragitem – Set Values in DRAGITEM

```
#define INCL_WINSTDDRAG
```

```
BOOL DrgSetDragitem (PDRAGINFO pDragInfo, PDRAGITEM pDragitem, ULONG cbBuffer,  
                    ULONG lItem)
```

This function sets the values in a DRAGITEM structure.

Parameters

pDragInfo (PDRAGINFO) – input
Pointer.

Pointer to the DRAGINFO structure in which to place the DRAGITEM.

pDragitem (PDRAGITEM) – input
Pointer.

Pointer to the DRAGITEM structure to place in DRAGINFO.

cbBuffer (ULONG) – input
DRAGITEM size.

Size of the DRAGITEM addressed by *pDragitem*.

lItem (ULONG) – input
DRAGITEM index.

Zero-based index of the DRAGITEM to be set.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Remarks

This function is used to initialize the DRAGINFO structure before calling DrgDrag.

This function is used only by the source of the drag, not by the target.

Related Functions

- DrgQueryDragitem

Example Code

This example shows a direct manipulation operation between two windows. The actual operation, copying the CONFIG.SYS file to C:\OS2\CONFIG.SYS, is visually represented by a drag and drop of an icon.

```
#define INCL_GPIBITMAPS /* GPI Bit Map Functions */
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#define INCL_DOSFILEMGR /* File Management Functions */
#define INCL_WININPUT /* Window Input Functions */
#include <os2.h>
#include <string.h>
#define ID_WINDOW 255
#define ID_ITEM 256
#define ID_BITMAP 257 /* .rc file: "bitmap 257 drgimage.bmp" */
HPS hps; /* Presentation space handle */
```

DrgSetDragitem – Set Values in DRAGITEM

```

BOOL      flResult;
HAB       hab;
PDRAGINFO pdinfo;
DRAGITEM  ditem;
DRAGIMAGE dimg;
PDRAGITEM pditem;
HBITMAP   hbm;           /* Bit-map handle          */
HPOINTER  hptr;          /* Pointer bit-map handle */
HWND      hwndDrop;
HWND      hwnd;
MPARAM     mpl;
char szBuffer[32];        /* Buffer where intersection string
                           /* is returned
char szSource[32];
char szTarget[32];

/*****
/* Inside ClientWindowProc of Source Window
*****/

case WM_BEGINDRAG:

/*****
/* Initialize the DRAGITEM structure
*****/

    ditem.hwndItem = hwnd;      /* Conversation partner */
    ditem.ulItemID = ID_ITEM;    /* Identifies item being dragged */
    ditem.hstrType = DrgAddStrHandle("DRT_TEXT"); /* Text item */
    ditem.hstrRMF = DrgAddStrHandle("<DRM_OS2FILE,DRF_TEXT>");
    ditem.hstrContainerName = DrgAddStrHandle("C:\\");
    ditem.hstrSourceName = DrgAddStrHandle("C:\\CONFIG.SYS");
    ditem.hstrTargetName = DrgAddStrHandle("C:\\OS2\\CONFIG.SYS");
    ditem.cxOffset = 0; ditem.cyOffset = 0;
    ditem.fsControl = 0; ditem.fsSupportedOps = 0;

/*****
/* Create the DRAGINFO structure
*****/

    pdinfo = DrgAllocDraginfo(1);

/*****
/* Initialize the DRAGIMAGE structure
*****/

    dimg.cb = sizeof(DRAGIMAGE); /* Size control block */
    dimg.cptl = 0;
    dimg.hImage = hbm;           /* Image handle passed to */
                                /* DrgDrag */
    dimg.sizlStretch.cx = 20L;   /* Size to stretch ico or */
    dimg.sizlStretch.cy = 20L;   /* bmp to */
    dimg.fl = DRG_BITMAP |
              DRG_STRETCH;        /* Stretch to size specified */
    dimg.cxOffset = 0;           /* Offset of the origin of the */
    dimg.cyOffset = 0;           /* image from the pointer */
                                /* hotspot */

    flResult= DrgSetDragitem(pdinfo, &ditem, (ULONG)sizeof(ditem), 0);

/*****
/* Perform the drag operation:
*****/

```


DrgSetDragitem — Set Values in DRAGITEM

```

        hwndDrop = DrgDrag(hwnd, /* Source of the drag          */
                           pdinfo, /* Pointer to DRAGINFO structure */
                           (PDRAGIMAGE)&dimg, /* Drag image          */
                           1, /* Size of the pdimg array */
                           VK_ENGDRAW, /* Release of drag button */
                           /* terminates the drag          */
                           NULL); /* Reserved          */

/*****
/* Inside ClientWindowProc of Target Window
*****/

case DM_DRAGOVER:

    pdinfo = MPFROMP(mp1);
    pditem = DrgQueryDragitemPtr(pdinfo,0);

    flResult = DrgVerifyTrueType(pditem,"DRF_TEXT");

    if(!flResult)

/*****
/* Inform the application that you will accept the drop
*****/

        return(MRFROM2SHORT(DOR_DROP,DO_COPY));

case DM_DROP:
    pdinfo = MPFROMP(mp1);
    pditem = DrgQueryDragitemPtr(pdinfo,0);

/*****
/* Perform the operation represented by the direct manipulation */
*****/

    DrgQueryStrName(pditem->hstrSourceName,sizeof(szSource),szSource);
    DrgQueryStrName(pditem->hstrTargetName,sizeof(szTarget),szTarget);
    flResult = DosCopy(szSource,szTarget,0L);

/*****
/* If operation is successful, return DMFL_TARGETSUCCESSFUL
*****/

    if(!flResult)
    {
        DrgSendTransferMsg(pditem->hwndItem,
                           DM_ENDCONVERSATION,
                           MPFROMLONG(pditem->ulItemID),
                           MPFROMLONG(DMFL_TARGETSUCCESSFUL));
    }

/*****
/* Otherwise, return DMFL_TARGETFAIL
*****/

else
{
    DrgSendTransferMsg(pditem->hwndItem,
                       DM_ENDCONVERSATION,
                       MPFROMLONG(pditem->ulItemID),
                       MPFROMLONG(DMFL_TARGETFAIL));
}

```

DrgSetDragPointer – Set Pointing Device Pointer

```
#define INCL_WINSTDDRAG
```

BOOL DrgSetDragPointer (PDRAGINFO pDragInfo, HPOINTER hptrHandle)

This function sets the pointer to be used while over the current target.

Parameters

pDragInfo (PDRAGINFO) – input
Pointer.

Pointer to the DRAGINFO structure to be used for this drag.

hpPtrHandle (HPOINTER) – input
Pointer handle.

Handle to the pointer to use.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INVALID_HPTR

An invalid pointer handle was specified.

Remarks

This function sets the pointer to be used to indicate the hot spot while dragging over the current target.

The pointer that is set with DrgSetDragPointer is used only while it is over the current target. The pointer is reset to the default when a new target is dragged over.

This function can be used by an application to provide meaningful augmentation emphasis for an operation that is unknown to the system (for example, swap).

When the drag pointer is successfully set, TRUE is returned.

Related Functions

- DrgSetDragImage

DrgSetDragPointer – Set Pointing Device Pointer

Example Code

This example uses the DrgSetDragPointer function to set the image used for the pointer while the pointer is over the target during a direct manipulation operation.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#include <OS2.H>
BOOL      f1Result;
PDRAGITEM pditem;
HPOINTER  hptrCrossHair;
MPARAM    mp1;
char szBuffer[32];

case DM_DRAGOVER:
    DrgSetDragPointer ((PDRAGINFO) mp1, hptrCrossHair);
```

DrgVerifyNativeRMF – Verify Native Rendering Mechanism and Format

```
#define INCL_WINSTDDRAG
```

BOOL DrgVerifyNativeRMF (PDRAGITEM pDragItem, PSZ pszRMF)

This function determines if the native rendering mechanism and format of an object match any supplied by the application.

Parameters

pDragItem (PDRAGITEM) – input
Pointer.

Pointer to the DRAGITEM structure whose native rendering mechanism and format are to be verified.

pszRMF (PSZ) – input
String.

A string specifying the rendering mechanism and format. The string is of the form: MECHFMT[,MECHFMT,MECHFMT,...], where MECHFMT can be in either of these formats:

- <mechanism(1),format(1)>
- (mechanism(1)[, mechanism(n)...]) (format(1)[,format(n)...])

Returns

Validity indicator.

TRUE Successful completion.

FALSE Error occurred.

Remarks

This function determines if the native rendering mechanism and format of a dragged object are understood by the target.

If TRUE is returned, the target may be able to carry out the action indicated by the direct manipulation itself, or it can select the native rendering mechanism and format as those to be used for the data exchange.

Related Functions

- DrgQueryNativeRMF

DrgVerifyNativeRMF —

Verify Native Rendering Mechanism and Format

Example Code

This example determines if the native rendering mechanism and format of an object match any supplied by the application.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#include <OS2.H>

DRAGITEM Dragitem;      /* DRAGITEM structure whose native
                        /* rendering mechanism and format are
                        /* to be verified

char pszRMF[25];         /* A string specifying the rendering
                        /* mechanism and format. The string is
                        /* of the form:
                        /*
                        /* mechfmt[,mechfmt,mechfmt,...],
                        /*
                        /* where 'mechfmt' can be in either of
                        /* these formats:
                        /*
                        /* o <mechanism(1),format(1)>
                        /* o (mechanism(1)[, mechanism(n)...])
                        /* (format(1)[,format(n)...])

strcpy(pszRMF,"(DRM_OS2FILE,DRF_TEXT)");
                        /* Mechanism is an OS/2 file and format
                        /* is a null-terminated string. See
                        /* the DRAGITEM structure for valid
                        /* formats.

if(DrgVerifyNativeRMF(&Dragitem, pszRMF))
{
    /* Code block
}
```

DrgVerifyRMF – Verify Given Rendering Mechanism and Format

```
#define INCL_WINSTDDRAG
```

BOOL DrgVerifyRMF (PDRAGITEM pDragItem, PSZ pszMech, PSZ pszFormat)

This function determines if a given rendering mechanism and format are supported for a dragged object.

Parameters

pDragItem (PDRAGITEM) – input
Pointer.

Pointer to the DRAGITEM structure whose native rendering mechanism and format are to be validated.

pszMech (PSZ) – input
Mechanism string.

A string specifying the rendering mechanism to search for. NULL will match any mechanism.

pszFormat (PSZ) – input
Format string.

A string specifying the rendering format to search for. NULL will match any format.

Returns

Validity indicator.

TRUE Successful completion.

FALSE Error occurred.

Remarks

This function determines if a given rendering mechanism and format ordered pair are represented in the set of valid pairs specified by *hstrRMF* for the dragged object.

Related Functions

- DrgVerifyNativeRMF

DrgVerifyRMF –

Verify Given Rendering Mechanism and Format

Example Code

This example determines if a given rendering mechanism and format are supported for a dragged object.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#include <OS2.H>

DRAGITEM Dragitem;      /* DRAGITEM structure whose native */
                        /* rendering mechanism and format are */
                        /* to be validated */
char pszMech[] = "DRM_OS2FILE";
                        /* A string specifying the rendering */
                        /* mechanism to search for */
char pszFormat[] = "DRF_TEXT";
                        /* A string specifying the rendering */
                        /* format to search for */

if(DrgVerifyRMF(&Dragitem, pszMech, pszFormat))
    /* Mechanism is an OS/2 file and format */
    /* is a null-terminated string */
{
    /* Code block */
}
```

DrgVerifyTrueType – Verify True Type of Dragged Object

```
#define INCL_WINSTDDRAG
```

BOOL DrgVerifyTrueType (PDRAGITEM pDragitem, PSZ pszType)

This function determines if the true type of a dragged object matches an application-supplied type string.

Parameters

pDragitem (PDRAGITEM) – input
Pointer.

Pointer to the DRAGITEM structure whose true type is to be verified.

pszType (PSZ) – input
Type string.

A string specifying a type. This string is in the format: TYPE[,TYPE...].

Returns

Validity indicator.

TRUE Successful completion.

FALSE Error occurred.

Remarks

If an item in the string pointed to by *pszType* matches the first type in the string associated with *hstrType* in the DRAGITEM structure, TRUE is returned.

A target application uses this function to determine if it supports the true type of a dragged object. If the application does not support the true type, it can either disallow a drop or change its default operation. If the default operation is a move, the drop should be disallowed, or the operation changed to a copy to prevent any loss of data for the object.

Related Functions

- DrgQueryTrueType
- DrgVerifyType
- DrgVerifyTypeSet

DrgVerifyTrueType — Verify True Type of Dragged Object

Example Code

This example verifies whether a given type is present in the list of types defined for a drag object.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#include <OS2.H>

BOOL      fValid;
DRAGITEM  Dragitem;      /* DRAGITEM structure whose hstrType is */
                          /* to be verified */

char pszType[8];          /* A string specifying the types to */
                          /* search for */

strcpy(pszType,"DRT_EXE"); /* Executable file type. See the */
                          /* DRAGINFO structure for valid */
                          /* types. */

fValid = DrgVerifyTrueType(&Dragitem, pszType);
```

DrgVerifyType – Verify Type of Dragged Object

```
#define INCL_WINSTDDRAG
```

BOOL DrgVerifyType (PDRAGITEM pDragitem, PSZ pszType)

This function verifies whether a given type is present in the list of types defined for a drag object.

Parameters

pDragitem (PDRAGITEM) – input
Pointer.

Pointer to the DRAGITEM structure whose *hstrType* is to be verified.

pszType (PSZ) – input
Type string.

A string specifying the types to search for. This string is in the format: TYPE[,TYPE...].

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INVALID_PARAMETERS

An application parameter value is invalid for its converted PM type. For example: a 4-byte value outside the range –32,768 to +32,767 cannot be converted to a SHORT, and a negative number cannot be converted to a ULONG or USHORT.

PMERR_INSUFFICIENT_MEMORY

The operation terminated through insufficient memory.

Remarks

If at least one of the types specified by *pszType* is present in *hstrType* in the DRAGITEM structure, TRUE is returned.

Related Functions

- DrgVerifyTrueType
- DrgVerifyTypeSet

DrgVerifyType — Verify Type of Dragged Object

Example Code

This example verifies whether a given type is present in the list of types defined for a drag object.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#include <OS2.H>

BOOL      fValid;
DRAGITEM Dragitem;      /* DRAGITEM structure whose hstrType is */
                        /* to be verified */
char pszType[] = "DRT_EXE";
                        /* A string specifying the types to */
                        /* search for */

fValid = DrgVerifyType(&Dragitem, pszType);
```

```
#define INCL_WINSTDDRAG
```

```
BOOL DrgVerifyTypeSet (PDRAGITEM pDragitem, PSZ pszType, ULONG cbBuflen,  
PSZ pszBuffer)
```

This function returns the intersection of the contents of the string associated with the type-string handle for an object and an application-specified type string.

Parameters

pDragItem (PDRAGITEM) – input
Pointer.

Pointer to the DRAGITEM structure whose *hstrType* is to be verified.

pszType (PSZ) – input
Type string.

A string specifying the types to search for. This string is in the format: TYPE[,TYPE...].

cbBuflen (ULONG) – input
Buffer size.

Size of the return buffer. The buffer should be at least one byte longer than the length of the string pointed to by *pszType*.

pszBuffer (PSZ) – output
Buffer.

Buffer where the intersection string is returned.

Returns

Match indicator.

TRUE Successful completion.

FALSE Error occurred.

Remarks

If at least one of the types specified by *pszType* is present in *hstrType* in the DRAGITEM structure, TRUE is returned.

Related Functions

- DrgVerifyType
- DrgVerifyTrueType

DrgVerifyTypeSet —

Verify Types

Example Code

In this example, the DrgVerifyTypeSet function is used to determine whether DRT_TEXT is among the types associated with the object. If it is, the drop is accepted.

```
#define INCL_WINSTDDRAG /* Direct Manipulation (Drag) Functions */
#include <OS2.H>
#include <stdio.h>
BOOL    flResult;
DRAGITEM pditem;
char    szBuffer[32];

case DM_DRAGOVER:

    flResult = DrgVerifyTypeSet(&pditem,
                                "DRT_TEXT",
                                sizeof(szBuffer),
                                szBuffer);

    flResult = strcmp(szBuffer,"DRT_TEXT");

    /******
    /* See if the object is an OS/2 file as well as being of text */
    /* format. AND result flag with previous result flag to get */
    /* the "effective" return code. */
    /******

    flResult = DrgVerifyRMF(&pditem,"DRM_OS2FILE","DRF_TEXT");

    /******
    /* See if DRT_TEXT is the true type of the object */
    /******

    flResult = DrgVerifyTrueType(&pditem,"DRF_TEXT");

    if(!flResult)

    /******
    /* Inform the application that you will accept the drop */
    /******

    return(MRFROM2SHORT(DOR_DROP, DO_COPY));
    break;
```

Chapter 4. Dynamic Data Formatting Functions

The following table shows all the dynamic data formatting (Ddf) functions in alphabetic order.

C Name
DdfBeginList
DdfBitmap
DdfEndList
DdfHyperText
DdfInform
DdfInitialize
DdfListItem
DdfMetafile
DdfPara
DdfSetColor
DdfSetFont
DdfSetFontStyle
DdfSetFormat
DdfSetTextAlign
DdfText

DdfBeginList – Begin Definition List

#define INCL_DDF

BOOL DdfBeginList (HDDF hddf, ULONG ulWidthDT, ULONG fBreakType, ULONG fSpacing)

This function begins a definition list in the DDF buffer; it corresponds to the :dl. (definition list) tag.

Parameters

hddf (HDDF) – input

Handle to DDF returned by DdfInitialize.

ulWidthDT (ULONG) – input

Width of the definition term.

fBreakType (ULONG) – input

Only the following constants may be specified:

HMBT_ALL	Start all definition descriptions on the next line, regardless of the actual lengths of definition terms.
HMBT_FIT	Start definition description on the next line only when the definition term is longer than the width specified.
HMBT_NONE	Do not start the definition description on the next line, even when the definition term is longer than the width specified.

fSpacing (ULONG) – input

Only the following constants may be specified:

HMLS_SINGLELINE	Do not insert a blank line between each definition description and the next definition term.
HMLS_DOUBLELINE	Insert a blank line between each definition description and the next definition term.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

HMERR_DDF_MEMORY	Not enough memory is available.
HMERR_DDF_LIST_UNCLOSED	An attempt was made to nest a list.
HMERR_DDF_LIST_BREAKTYPE	The value of BreakType is not valid.
HMERR_DDF_LIST_SPACING	The value for Spacing is not valid.

Remarks

Once this function has been called, use of any DDF function other than DdfListItem, DdfSetColor, and DdfEndList may produce unpredictable results.

DdfBeginList – Begin Definition List

Related Functions

- DdfText
- DdfSetTextAlign
- DdfSetFormat
- DdfSetFontStyle
- DdfSetFont
- DdfSetColor
- DdfPara
- DdfMetafile
- DdfListItem
- DdfInitialize
- DdfInform
- DdfHyperText
- DdfEndList
- DdfBitmap

Example Code

After initializing a DDF buffer with DdfInitialize, the example uses DdfBeginList to indicate the beginning of a definition list in the DDF buffer (this corresponds to the IPF dl tag). For a more detailed example and discussion of initializing DDF, see the DdfInitialize sample.

```
#define INCL_WINWINDOWMGR      /* General window management */
#define INCL_WINMESSAGEMGR    /* Message management */
#define INCL_DDF               /* Dynamic Data Facility */
#include <os2.h>
#include <pmhelp.h>

struct _LISTITEM              /* definition list */
{
    PSZ Term;
    PSZ Desc;
} Definition[2] = {{ "MVS", "Multiple Virtual
System"},
                  { "VM", "Virtual Machine" }};
MRESULT WindowProc( HWND hwnd, ULONG ulMsg, MPARAM mp1, MPARAM mp2)
{
    HWND    hwndParent;
    HWND    hwndInstance;
    Hddf    hDdf;          /* DDF handle */
    SHORT i;               /* loop index */

    switch( ulMsg )
    {
    case HM_QUERY_DDF_DATA:
        /* get the help instance */
        hwndParent = WinQueryWindow( hwnd, QW_PARENT );
        hwndParent = WinQueryWindow( hwndParent, QW_PARENT );
        hwndInstance = (HWND)WinSendMsg( hwndParent, HM_QUERY,
                                          MPFROMSHORT( HMQW_INSTANCE ), NULL );

        /* Allocate 1K Buffer (default) */
        hDdf = DdfInitialize(
            hwndInstance, /* Handle of help instance */
            0L,          /* Default buffer size */
            0L           /* Default increment */
        );
    }
```


DdfBeginList – Begin Definition List

```
    if (hDdf == NULLHANDLE)      /* Check return code      */
    {
        return (MRESULT)FALSE;
    }

    /* begin definition list */
    if (!DdfBeginList(hDdf, 3L, HMBT_ALL, HMLS_SINGLELINE))
    {
        return (MRESULT)FALSE;
    }

    /* insert 2 entries into definition list */
    for (i=0; i < 2; i++)
    {
        if (!DdfListItem(hDdf, Definition[i].Term,
                          Definition[i].Desc))
        {
            return (MRESULT)FALSE;
        }
    }

    /* terminate definition list */
    if (!DdfEndList(hDdf))
    {
        return (MRESULT)FALSE;
    }

    return (MRESULT)hDdf;
}
}
```

DdfBitmap – Place Bitmap Reference

```
#define INCL_DDF
```

BOOL DdfBitmap (HDDF hddf, HBITMAP hbm, ULONG fAlign)

This function places a reference to a bit map in the DDF buffer.

Parameters

hddf (HDDF) – input

Handle to DDF returned by DdfInitialize.

hbm (HBITMAP) – input

Standard Presentation Manager bit map handle.

fAlign (ULONG) – input

Any of the following values can be specified:

ART_LEFT	Left-justify the bit map.
ART_RIGHT	Right-justify the bit map.
ART_CENTER	Center the bit map.
ART_RUNIN	Allow the bit map to be reflowed with text.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

HMERR_DDF_MEMORY Not enough memory is available.

HMERR_DDF_ALIGN_TYPE The alignment type is not valid.

Remarks

The handle to the presentation space in which the bit map was created cannot be freed by the application while the panel is displayed.

Note: There is a (3-byte + size of HBITMAP structure) ESC code overhead in the DDF internal buffer for this function. There is a 1-byte ESC code overhead required for the Align flag.

Related Functions

- DdfText
- DdfSetTextAlign
- DdfSetFormat
- DdfSetFontStyle
- DdfSetFont
- DdfSetColor
- DdfPara
- DdfMetafile
- DdfListItem

DdfBitmap — Place Bitmap Reference

- DdfInitialize
- DdfInform
- DdfHyperText
- DdfEndList
- DdfBeginList

Example Code

After initializing a DDF buffer with DdfInitialize, the example obtains a device context (DevOpenDC), creates a presentation space (GpiCreatePS), and loads a bit map (GpiLoadBitmap). It then uses DdfBitmap to place a reference to the bit map in the DDF buffer. For a more detailed example and discussion of initializing DDF, see the DdfInitialize sample.

```
#define INCL_WINWINDOWMGR      /* General window management */
#define INCL_WINMESSAGEMGR     /* Message management */
#define INCL_GPICONTROL        /* Basic PS control */
#define INCL_GPIBITMAPS        /* Bit maps and Pel Operations */
#define INCL_GPIPRIMITIVES     /* Drawing Primitives/Attributes*/
#define INCL_DDF               /* Dynamic Data Facility */
#include <os2.h>
#include <pmhelp.h>

#define ACVP_HAB 12
#define BM_HPS 16
#define BM_HDC 20
#define BM_HWND 24
#define ID_LEFT 255

MRESULT WindowProc( HWND hwnd, ULONG ulMsg, MPARAM mp1, MPARAM mp2 )
{
    HWND    hwndParent;      /* parent window */
    HWND    hwndInstance;    /* help instance window */
    Hddf     hDdf;           /* DDF handle */
    HDC      hdc;            /* device context handle */
    HPS      hps;            /* presentation space handle */
    HAB      hab;            /* anchor block handle */
    SIZEL    szel = {0L,0L}; /* size of new PS */
    HBITMAP  hBitmap;        /* bit map handle */
    HMODULE  hModule;        /* module handle */

    switch( ulMsg )
    {
        case HM_QUERY_DDF_DATA:
            hwndParent = WinQueryWindow( hwnd, QW_PARENT );
            hwndParent = WinQueryWindow( hwndParent, QW_PARENT );
            hwndInstance = (HWND)WinSendMsg( hwndParent, HM_QUERY,
                MPFROMSHORT( HMQW_INSTANCE ), NULL );

            /* Allocate 1K Buffer (default) */
            hDdf = DdfInitialize(
                hwndInstance, /* Handle of help instance */
                0L,          /* Default buffer size */
                0L,          /* Default increment */
            );

            if (hDdf == NULLHANDLE) /* Check return code */
            {
                return (MRESULT)FALSE;
            }

            /* get module handle for bit map */
            DosGetModHandle("bitmap", &hModule);
            if (hModule == NULLHANDLE)
            {
```

DdfBitmap – Place Bitmap Reference

```

        return (MRESULT)FALSE;
    }

    /* get hab for this window */
    if ((hab = (HAB)WinQueryWindowULong(hwnd, ACVP_HAB)) == NULLHANDLE)
    {
        return (MRESULT)FALSE;
    }

    /* create a device context */
    if ((hdc = DevOpenDC(hab, OD_MEMORY, "", 0L,
                        (PDEVOPENDATA)NULL, (HDC)NULL)) == NULLHANDLE)
    {
        return (MRESULT)FALSE;
    }

    /* save hdc in reserved word */
    WinSetWindowULong(hwnd, BM_HDC, (ULONG)hdc);

    /* create a noncached micro presentation space */
    /* and associate it with the window */
    if ((hps = GpiCreatePS(hab, hdc, &size1, PU_PELS |
GPIF_DEFAULT
                                | GPIT_MICRO | GPIA_ASSOC)) == NULLHANDLE)
    {
        return (MRESULT)FALSE;
    }

    /* save hps in reserved word */
    WinSetWindowULong(hwnd, BM_HPS, (ULONG)hps);

    /* Load the Bit map to display */
    if ((hBitmap = GpiLoadBitmap(hps, hModule, ID_LEFT, 300L,
                                300L)) == NULLHANDLE)
    {
        return (MRESULT)FALSE;
    }

    /* save bit map hwnd in reserved word */
    WinSetWindowULong(hwnd, BM_HWND, (ULONG)hBitmap);

    /* Display the bit map align left */
    if (!DdfBitmap(hDdf, hBitmap, (ULONG)TA_LEFT))
    {
        return (MRESULT)FALSE;
    }

    return (MRESULT)hDdf;

case WM_CLOSE:
    /* release PS, DC, and bit map */
    GpiDestroyPS((HPS)WinQueryWindowULong(hwnd, BM_HPS));
    DevCloseDC((HDC)WinQueryWindowULong(hwnd, BM_HDC));
    GpiDeleteBitmap((HBITMAP)WinQueryWindowULong(hwnd, BM_HWND));
    WinDestroyWindow(WinQueryWindow(hwnd, QW_PARENT));
    return (MRESULT)TRUE;
}
}

```

DdfEndList — End Definition List

```
#define INCL_DDF
```

BOOL DdfEndList (HDDF hddf)

This function terminates the definition list initialized by DdfBeginList.

Parameters

hddf (HDDF) — input

Handle to DDF returned by DdfInitialize.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

HMERR_DDF_LIST_UNINITIALIZED No definition list has been initialized by DdfBeginList.

Related Functions

- DdfText
- DdfSetTextAlign
- DdfSetFormat
- DdfSetFontStyle
- DdfSetFont
- DdfSetColor
- DdfPara
- DdfMetafile
- DdfListItem
- DdfInitialize
- DdfInform
- DdfHyperText
- DdfBitmap
- DdfBeginList

Example Code

After initializing a DDF buffer with DdfInitialize, the example uses DdfBeginList to indicate the beginning of a definition list in the DDF buffer (this corresponds to the IPF dl tag). For a more detailed example and discussion of initializing DDF, see the DdfInitialize sample.

```
#define INCL_WINWINDOWMGR      /* General window management */
#define INCL_WINMESSAGEMGR    /* Message management */
#define INCL_DDF               /* Dynamic Data Facility */
#include <os2.h>
#include <pmhelp.h>

struct _LISTITEM              /* definition list */
{
    PSZ Term;
```

DdfEndList – End Definition List

```

    PSZ Desc;
} Definition[2] = {"MVS", "Multiple Virtual
System"},
                {"VM", "Virtual Machine"}};
MRESULT WindowProc( HWND hwnd, ULONG uMsg, MPARAM mp1, MPARAM mp2)
{
    HWND  hwndParent;
    HWND  hwndInstance;
    Hddf  hDdf;          /* DDF handle          */
    SHORT i;             /* loop index          */

    switch( uMsg )
    {
    case HM_QUERY_DDF_DATA:
        /* get the help instance */
        hwndParent = WinQueryWindow( hwnd, QW_PARENT );
        hwndParent = WinQueryWindow( hwndParent, QW_PARENT );
        hwndInstance = (HWND)WinSendMsg( hwndParent, HM_QUERY,
                                         MPFROMSHORT( HMQW_INSTANCE ), NULL );

        /* Allocate 1K Buffer (default) */
        hDdf = DdfInitialize(
            hwndInstance, /* Handle of help instance */
            0L,           /* Default buffer size     */
            0L            /* Default increment       */
        );

        if (hDdf == NULLHANDLE) /* Check return code */
        {
            return (MRESULT)FALSE;
        }

        /* begin definition list */
        if (!DdfBeginList(hDdf, 3L, HMBT_ALL, HMLS_SINGLELINE))
        {
            return (MRESULT)FALSE;
        }

        /* insert 2 entries into definition list */
        for (i=0; i < 2; i++)
        {
            if (!DdfListItem(hDdf, Definition[i].Term,
                             Definition[i].Desc))
            {
                return (MRESULT)FALSE;
            }
        }

        /* terminate definition list */
        if (!DdfEndList(hDdf))
        {
            return (MRESULT)FALSE;
        }

        return (MRESULT)hDdf;
    }
}

```

DdfHyperText – Define Hypertext Link

```
#define INCL_DDF
```

BOOL DdfHyperText (HDDF hddf, PSZ pszText, PSZ pszReference, ULONG fReferenceType)

This function defines a hypertext link to another panel.

Parameters

hddf (HDDF) – input

Handle to DDF returned by DdfInitialize.

pszText (PSZ) – input

Hypertext phrase.

pszReference (PSZ) – input

The value of this parameter depends on the value of ReferenceType:

- If ReferenceType is REFERENCE_BY_RES, this parameter must contain a pointer to a numeric string containing the res number; otherwise it will default to a res number of zero. Valid values are 1 - 64000; all other values are reserved.
- If ReferenceType is REFERENCE_BY_ID, this parameter contains a pointer to a string containing the alphanumeric identifier of the destination panel.

fReferenceType (ULONG) – input

This parameter specifies whether you are linking via a resource identifier (res number) or via an alphanumeric identifier.

REFERENCE_BY_RES To link via a resource identifier.

REFERENCE_BY_ID To link via an alphanumeric identifier.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

HMERR_DDF_MEMORY Not enough memory is available.

HMERR_DDF_REFTYPE The reference type is not valid.

Remarks

Note: There is a 3-byte ESC code overhead in the DDF internal buffer for each word in the text buffer. There is a 1-byte ESC code overhead for each blank and for each newline character. If ReferenceType is REFERENCE_BY_ID, then there is a (3-byte + Reference length) ESC code overhead. For a ReferenceType of REFERENCE_BY_RES, the overhead is 5 bytes. Finally, there is a 3-byte ESC code overhead that is required for ending the hypertext link.

Related Functions

- DdfText
- DdfSetTextAlign
- DdfSetFormat
- DdfSetFontStyle
- DdfSetFont
- DdfSetColor
- DdfPara
- DdfMetafile
- DdfListItem
- DdfInitialize
- DdfInform
- DdfEndList
- DdfBitmap
- DdfBeginList

Example Code

After initializing a DDF buffer with DdfInitialize, the example uses DdfHyperText to create a hypertext link with another resource. For a more detailed example and discussion of initializing DDF, see the DdfInitialize sample.

```
#define INCL_WINWINDOWMGR      /* General window management */
#define INCL_WINMESSAGEGR     /* Message management */
#define INCL_DDF               /* Dynamic Data Facility */
#include <os2.h>
#include <pmhelp.h>

PSZ   Text = "This text is a HYPERTEXT message.\n"; /* hypertext
                                                    string */
PSZ   ResID = "1"; /* Resource identifier */

MRESULT WindowProc( HWND hwnd, ULONG ulMsg, MPARAM mp1, MPARAM mp2 )
{
    HWND   hwndParent;
    HWND   hwndInstance;
    Hddf   hDdf; /* DDF handle */

    switch( ulMsg )
    {
        case HM_QUERY_DDF_DATA:
            /* get the help instance */
            hwndParent = WinQueryWindow( hwnd, QW_PARENT );
            hwndParent = WinQueryWindow( hwndParent, QW_PARENT );
            hwndInstance = (HWND)WinSendMsg( hwndParent, HM_QUERY,
                                            MPFROMSHORT( HMQW_INSTANCE ), NULL );
```


DdfHyperText — Define Hypertext Link

```
/* Allocate 1K Buffer (default) */
hDdf = DdfInitialize(
    hwndInstance, /* Handle of help instance */
    0L,           /* Default buffer size */
    0L,           /* Default increment */
);

if (hDdf == NULLHANDLE) /* Check return code */
{
    return (MRESULT)FALSE;
}

/* create hypertext link with resource 1 */
if (!DdfHyperText(hDdf, (PSZ)Text, ResID, REFERENCE_BY_RES))
{
    return (MRESULT)FALSE;
}

return (MRESULT)hDdf;
}
```

```
#define INCL_DDF
```

BOOL DdfInform (HDDF hddf, PSZ pszText, ULONG resInformNumber)

This function defines a hypertext inform link; it corresponds to the link tag with reftype=inform.

Parameters

hddf (HDDF) — input

Handle to DDF returned by DdfInitialize.

pszText (PSZ) — input

Hypertext phrase.

resInformNumber (ULONG) — input

Res number associated with this hypertext field. Possible values are 1 to 64000; all other values are reserved.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

HMERR_DDF_MEMORY

Not enough memory is available.

Related Functions

- DdfText
- DdfSetTextAlign
- DdfSetFormat
- DdfSetFontStyle
- DdfSetFont
- DdfSetColor
- DdfPara
- DdfMetafile
- DdfListItem
- DdfInitialize
- DdfHyperText
- DdfEndList
- DdfBitmap
- DdfBeginList

Example Code

After initializing a DDF buffer with DdfInitialize, the example uses DdfInform to create a hypertext inform link with another resource (corresponds to the IPF :link. tag with reftype=inform). For a more detailed example and discussion of initializing DDF, see the DdfInitialize sample.

```
#define INCL_WINWINDOWMGR    /* General window management */
#define INCL_WINMESSAGEGR    /* Message management */
#define INCL_DDF              /* Dynamic Data Facility */
#include <os2.h>
#include <pmhelp.h>
```

DdfInform —

Define Inform Link

```
PSZ    Text = "This text is a HYPERTEXT message.\n"; /* hypertext
                                                    string */
MRESULT WindowProc( HWND hwnd, ULONG uMsg, MPARAM mp1, MPARAM mp2 )
{
    HWND  hwndParent;
    HWND  hwndInstance;
    Hddf  hDdf;          /* DDF handle */

    switch( uMsg )
    {
        case HM_QUERY_DDF_DATA:
            /* get the help instance */
            hwndParent = WinQueryWindow( hwnd, QW_PARENT );
            hwndParent = WinQueryWindow( hwndParent, QW_PARENT );
            hwndInstance = (HWND)WinSendMsg( hwndParent, HM_QUERY,
                                             MPFROMSHORT( HMQW_INSTANCE ), NULL );

            /* Allocate 1K Buffer (default) */
            hDdf = DdfInitialize(
                hwndInstance, /* Handle of help instance */
                0L,          /* Default buffer size */
                0L,          /* Default increment */
            );

            if (hDdf == NULLHANDLE) /* Check return code */
            {
                return (MRESULT)FALSE;
            }

            /* create hypertext inform link with resource 1 */
            if (!DdfInform(hDdf, (PSZ)Text, 1L))
            {
                return (MRESULT)FALSE;
            }

            return (MRESULT)hDdf;
        }
    }
}
```

DdfInitialize – Initialize DDF Area

```
#define INCL_DDF
```

HDDF DdfInitialize (HWND hwndHelpInstance, ULONG cbBuffer, ULONG ullIncrement)

This function initializes the IPF internal structures for dynamic data formatting and returns a DDF handle. The application uses this handle to refer to a particular DDF panel.

Parameters

hwndHelpInstance (HWND) – input

Handle of a help instance.

cbBuffer (ULONG) – input

Initial length of internal buffer where DDF information is to be stored. If this field is NULL, a default value of 1K is defined. The maximum value is 60KB.

ullIncrement (ULONG) – input

Amount by which to increment the buffer size, if necessary. If this field is NULL, a default value of 256 bytes is defined. The maximum value is 60KB.

Returns

A handle to DDF (HDDF) is returned if initialization was successful. Otherwise, the value returned is:

NULL An error has occurred because of insufficient memory or incorrect instance.

Remarks

At initialization, the default for dynamic data display is that text aligned on the left, and formatting is turned on.

Related Functions

- DdfText
- DdfSetTextAlign
- DdfSetFormat
- DdfSetFontStyle
- DdfSetFont
- DdfSetColor
- DdfPara
- DdfMetafile
- DdfListItem
- DdfInform
- DdfHyperText
- DdfEndList
- DdfBitmap
- DdfBeginList

Example Code

This example shows how to initialize and use the Dynamic Data Facility for displaying an online document. Two functions are defined: the first, SampleObj, creates a window that will display the online information and specifies the second function, SampleWindowProc, as the corresponding window procedure. These two functions are compiled into a DLL and exported, so that IPF can invoke them when it encounters the :ddf and :acviewport tags during execution. The :acviewport tag will specify the DLL name and the SampleObj function; when IPF calls SampleObj, it initializes an

DdfInitialize — Initialize DDF Area

application-controlled window with SampleWindowProc as the window procedure and returns the window handle. Later, when IPF encounters the :ddf tag, it will send SampleWindowProc an HM_QUERY_DDF_DATA message. At this point, before calling any of the DDF API, DdfInitialize must first be called to initiate a DDF buffer, after which the other DDF API can be called to display the online information.

```
#define INCL_WINWINDOWMGR      /* General window management */
#define INCL_WINMESSAGEMGR    /* Message management */
#define INCL_WINDIALOGS       /* Dialog boxes */
#define INCL_DDF               /* Dynamic Data Facility */
#define INCL_32
#include <os2.h>
#include <pmhelp.h>

#define COM_HWND 4              /* window word offsets */
#define PAGE_HWND 8
#define ACVP_HAB 12

USHORT DdfClass = FALSE;

MRESULT EXPENTRY SampleWindowProc(HWND hWnd, ULONG Message,
                                   MPARAM lParam1, MPARAM lParam2);

USHORT APIENTRY SampleObj(PACVP pACVP, PCH Parameter)
{
    HWND DdfHwnd;              /* Client window handle */
    HWND DdfCHwnd;             /* Child window handle */
    HWND PreviousHwnd;         /* Handle for setting comm window active */

    /* register DDF Base class if not registered already */
    if (!DdfClass)
    {
        if (!WinRegisterClass(
            pACVP->hAB,         /* Anchor block handle */
            "CLASS_Ddf",       /* Application window class name */
            SampleWindowProc,   /* Address of window procedure */
            CS_SYNCPAINT | CS_SIZEREDRAW | CS_MOVENOTIFY, /* Window class style */
            20))                /* Extra storage */
        {
            return TRUE;
        }
        DdfClass = TRUE;
    }

    /* create standard window */
    if (!(DdfHwnd = WinCreateStdWindow(
        pACVP->hWndParent,      /* ACVP is parent */
        0L,                    /* No class style */
        NULL,                   /* Frame control flag */
        "CLASS_Ddf",           /* Window class name */
        NULL,                   /* No title bar */
        0L,                    /* No special style */
        0L,                     /* Resource in .EXE */
        0,                      /* No window identifier */
        &DdfCHwnd )))           /* Client window handle */
    {
        return FALSE;
    }

    /* store the frame window handle in ACVP data structure */
    pACVP->hWndACVP = DdfHwnd;
```

DdfInitialize – Initialize DDF Area

```

/* set this window as active communication window */
PreviousHwnd = (HWND)WinSendMessage(pACVP->hWndParent,
                                     HM_SET_OBJCOM_WINDOW,
                                     MPFROMHWND(DdfHwnd), NULL);

/* save returned communication hwnd in reserved word */
WinSetWindowULong(DdfCHwnd, COM_HWND, (ULONG)PreviousHwnd);

/* save anchor block handle in reserved word */
WinSetWindowULong(DdfCHwnd, ACVP_HAB, (ULONG)pACVP->hAB);

return FALSE;
} /* SampleObj */

MRESULT EXPENTRY SampleWindowProc(HWND hWnd, ULONG Message,
                                   MPARAM lParam1, MPARAM lParam2)
{
    HWND  hwndParent;      /* parent window */
    HWND  hwndInstance;    /* help instance window */
    Hddf  hDdf;            /* DDF handle */
    ULONG DdfID;           /* DDF resource id */

    switch (Message)
    {
        case HM_QUERY_DDF_DATA:
            WinSetWindowULong(hWnd, PAGE_HWND, LONGFROMMP(lParam1));
            DdfID = LONGFROMMP(lParam2);
            hwndParent = WinQueryWindow(hWnd, QW_PARENT);
            hwndParent = WinQueryWindow(hwndParent, QW_PARENT);
            hwndInstance = (HWND)WinSendMessage(hwndParent, HM_QUERY,
                                                MPFROMSHORT(HMQW_INSTANCE), NULL);

            /* Allocate 1K Buffer (default) */
            hDdf = DdfInitialize(
                hwndInstance, /* Handle of help instance */
                0L,          /* Default buffer size */
                0L           /* Default increment */
            );

            if (hDdf == NULLHANDLE) /* Check return code */
            {
                return (MRESULT)FALSE;
            }

            return (MRESULT)hDdf;

        default:
            return (WinDefWindowProc(hWnd, Message, lParam1, lParam2));
    }
} /* SampleWindowProc */

```

DdfListItem – Insert List Item

```
#define INCL_DDF
```

BOOL DdfListItem (HDDF hddf, PSZ pszTerm, PSZ pszDescription)

This function inserts a definition list entry in the DDF buffer; it corresponds to a combination of the :dt. (definition term) and :dd. (definition define) tags.

Parameters

hddf (HDDF) – input

Handle to DDF returned by DdfInitialize.

pszTerm (PSZ) – input

Term portion of the definition list entry.

pszDescription (PSZ) – input

Description portion of the definition list entry.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

HMERR_DDF_MEMORY

Not enough memory is available.

HMERR_DDF_LIST_UNINITIALIZED

No definition list has been initialized by DdfBeginList.

Remarks

The handle to the presentation space in which the bit map was created cannot be freed by the application while the panel is displayed.

Note: There is a (3-byte + size of HBITMAP structure) ESC code overhead in the DDF internal buffer for this function. There is a 1-byte ESC code overhead required for the Align flag.

Related Functions

- DdfText
- DdfSetTextAlign
- DdfSetFormat
- DdfSetFontStyle
- DdfSetFont
- DdfSetColor
- DdfPara
- DdfMetafile
- DdfInitialize
- DdfInform
- DdfHyperText
- DdfEndList
- DdfBitmap
- DdfBeginList

Example Code

After initializing a DDF buffer with `DdfInitialize`, the example uses `DdfBeginList` to indicate the beginning of a definition list in the DDF buffer (this corresponds to the IPF dl tag). For a more detailed example and discussion of initializing DDF, see the `DdfInitialize` sample.

```
#define INCL_WINWINDOWMGR      /* General window management */
#define INCL_WINMESSAGEMGR    /* Message management */
#define INCL_DDF               /* Dynamic Data Facility */
#include <os2.h>
#include <pmhelp.h>

struct _LISTITEM              /* definition list */
{
    PSZ Term;
    PSZ Desc;
} Definition[2] = {"MVS", "Multiple Virtual
System"},
                {"VM", "Virtual Machine"};
MRESULT WindowProc( HWND hwnd, ULONG ulMsg, MPARAM mp1, MPARAM mp2)
{
    HWND  hwndParent;
    HWND  hwndInstance;
    Hddf  hDdf;          /* DDF handle */
    SHORT i;             /* loop index */

    switch( ulMsg )
    {
        case HM_QUERY_DDF_DATA:
            /* get the help instance */
            hwndParent = WinQueryWindow( hwnd, QW_PARENT );
            hwndParent = WinQueryWindow( hwndParent, QW_PARENT );
            hwndInstance = (HWND)WinSendMsg( hwndParent, HM_QUERY,
                MPFROMSHORT( HMQW_INSTANCE ), NULL );

            /* Allocate 1K Buffer (default) */
            hDdf = DdfInitialize(
                hwndInstance, /* Handle of help instance */
                0L,          /* Default buffer size */
                0L           /* Default increment */
            );

            if (hDdf == NULLHANDLE) /* Check return code */
            {
                return (MRESULT)FALSE;
            }

            /* begin definition list */
            if (!DdfBeginList(hDdf, 3L, HMBT_ALL, HMLS_SINGLELINE))
            {
                return (MRESULT)FALSE;
            }
    }
}
```


DdfListItem — Insert List Item

```
/* insert 2 entries into definition list */
for (i=0; i < 2; i++)
{
    if (!DdfListItem(hDdf, Definition[i].Term,
                    Definition[i].Desc))
    {
        return (MRESULT)FALSE;
    }
}

/* terminate definition list */
if (!DdfEndList(hDdf))
{
    return (MRESULT)FALSE;
}

return (MRESULT)hDdf;
} }
```

DdfMetafile – Place Metafile Reference

```
#define INCL_DDF
```

```
BOOL DdfMetafile (HDDF hddf, HMF hmf, PRECTL prclRect)
```

This function places a reference to a metafile into the DDF buffer.

Parameters

hddf (HDDF) – input

Handle to DDF returned by DdfInitialize.

hmf (HMF) – input

The handle of the metafile to display.

prclRect (PRECTL) – input

If not NULL, contains the size of the rectangle in which the metafile will be displayed. The aspect ratio of the metafile is adjusted to fit this rectangle.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

HMERR_DDF_MEMORY Not enough memory is available.

Remarks

Note: There is a 3-byte ESC code overhead in the DDF internal buffer for this function. There is also a (MetaFilename length) overhead. Finally, the Rect variable requires an additional 16 bytes of overhead in the DDF internal buffer.

Related Functions

- DdfText
- DdfSetTextAlign
- DdfSetFormat
- DdfSetFontStyle
- DdfSetFont
- DdfSetColor
- DdfPara
- DdfListItem
- DdfInitialize
- DdfInform
- DdfHyperText
- DdfEndList
- DdfBitmap
- DdfBeginList

DdfMetafile — Place Metafile Reference

Example Code

After initializing a DDF buffer with DdfInitialize and loading a metafile with GpiLoadMetaFile, the example uses DdfMetafile to place a reference to the metafile in the DDF buffer. For a more detailed example and discussion of initializing DDF, see the DdfInitialize sample.

```
#define INCL_WINWINDOWMGR      /* General window management */
#define INCL_WINMESSAGEMGR    /* Message management */
#define INCL_DDF              /* Dynamic Data Facility */
#define INCL_GPIMETAFILES     /* MetaFiles */
#include <os2.h>
#include <pmhelp.h>

#define MF_HWND 0
#define ACVP_HAB 4

MRESULT WindowProc( HWND hwnd, ULONG ulMsg, MPARAM mp1, MPARAM mp2 )
{
    HWND  hwndParent;
    HAB   hab;
    HWND  hwndInstance; /* help instance window */
    Hddf  hDdf; /* DDF handle */
    HMF   hwndMetaFile; /* metafile handle */

    switch( ulMsg )
    {
        case HM_QUERY_DDF_DATA:
            /* get the help instance */
            hwndParent = WinQueryWindow( hwnd, QW_PARENT );
            hwndParent = WinQueryWindow( hwndParent, QW_PARENT );
            hwndInstance = (HWND)WinSendMsg( hwndParent, HM_QUERY,
                MPFROMSHORT( HMQW_INSTANCE ), NULL );

            /* Allocate 1K Buffer (default) */
            hDdf = DdfInitialize(
                hwndInstance, /* Handle of help instance */
                0L,          /* Default buffer size */
                0L,          /* Default increment */
            );

            if (hDdf == NULLHANDLE) /* Check return code */
            {
                return (MRESULT)FALSE;
            }

            /* get hab for this window */
            if ((hab = (HAB)WinQueryWindowULong(hwnd, ACVP_HAB)) == NULLHANDLE)
            {
                return (MRESULT)FALSE;
            }

            /* Load the Metafile to display */
            if ((hwndMetaFile = GpiLoadMetaFile(hab, "SAMP.MET")) == NULLHANDLE)
            {
                return (MRESULT)FALSE;
            }
    }
}
```

DdfMetafile — Place Metafile Reference

```
/* Save MetaFile hwnd in reserved word */
WinSetWindowULong(hwnd, MF_HWND, hwndMetaFile);

if (!DdfMetafile(hDdf, hwndMetaFile, NULL))
{
    return (MRESULT)FALSE;
}

return (hDdf);

case WM_CLOSE:
    GpiDeleteMetaFile((HMF)WinQueryWindowULong(hwnd, MF_HWND));
    WinDestroyWindow(WinQueryWindow(hwnd, QW_PARENT));

    return (MRESULT)TRUE;
}
return WinDefWindowProc( hwnd, ulMsg, mp1, mp2 );
}
```

DdfPara — Create a Paragraph in DDF Buffer

```
#define INCL_DDF
```

```
BOOL DdfPara (HDDF hddf)
```

This function creates a paragraph within the DDF buffer. It corresponds to the `:p.` tag. This function places a reference to a bit map in the DDF buffer.

Parameters

hddf (HDDF) — input

Handle to DDF returned by `DdfInitialize`.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from `WinGetLastError`

HMERR_DDF_MEMORY

Not enough memory is available.

Remarks

Note: There is a 1-byte ESC code overhead in the DDF internal buffer for this function.

Related Functions

- `DdfText`
- `DdfSetTextAlign`
- `DdfSetFormat`
- `DdfSetFontStyle`
- `DdfSetFont`
- `DdfSetColor`
- `DdfMetafile`
- `DdfListItem`
- `DdfInitialize`
- `DdfInform`
- `DdfHyperText`
- `DdfEndList`
- `DdfBitmap`
- `DdfBeginList`

Example Code

After initializing a DDF buffer with `DdfInitialize`, the example uses `DdfPara` to start a new paragraph, `DdfSetFont` and `DdfSetFontStyle` to have the text displayed in a large, bold Courier font, `DdfSetColor` to change the text color, and `DdfText` to place text in the buffer. For a more detailed example and discussion of initializing DDF, see the `DdfInitialize` sample.

```
#define INCL_WINWINDOWMGR    /* General window management */
#define INCL_WINMESSAGEMGR  /* Message management */
#define INCL_DDF             /* Dynamic Data Facility */
#include <os2.h>
#include <pmhelp.h>
```

DdfPara — Create a Paragraph in DDF Buffer

```

MRESULT WindowProc( HWND hwnd, ULONG ulMsg, MPARAM mp1, MPARAM mp2 )
{
    HWND    hwndParent;
    HWND    hwndInstance;    /* help instance window */
    Hddf    hDdf;            /* DDF handle */

    switch( ulMsg )
    {
        case HM_QUERY_DDF_DATA:
            /* get the help instance */
            hwndParent = WinQueryWindow( hwnd, QW_PARENT );
            hwndParent = WinQueryWindow( hwndParent, QW_PARENT );
            hwndInstance = (HWND)WinSendMsg( hwndParent, HM_QUERY,
                                             MPFROMSHORT( HMQW_INSTANCE ), NULL );

            /* Allocate 1K Buffer (default) */
            hDdf = DdfInitialize(
                hwndInstance, /* Handle of help instance */
                0L,           /* Default buffer size */
                0L           /* Default increment */
            );

            if (hDdf == NULLHANDLE)    /* Check return code */
            {
                return (MRESULT)FALSE;
            }

            /* create paragraph in DDF buffer */
            if( !DdfPara( hDdf ) )
            {
                return (MRESULT)FALSE;
            }

            /* Change to large (100 x 100 dimensions) Courier font */
            if( !DdfSetFont( hDdf, "Courier", 100L, 100L ) )
            {
                return (MRESULT)FALSE;
            }

            /* make the font BOLDFACE */
            if( !DdfSetFontStyle( hDdf, FM_SEL_BOLD ) )
            {
                return (MRESULT)FALSE;
            }

            /* make the text display as BLUE on a PALE GRAY background */
            if( !DdfSetColor( hDdf, CLR_PALEGRAY, CLR_BLUE ) )
            {
                return (MRESULT)FALSE;
            }

            /* Write data into the buffer */
            if ( !DdfText(hDdf, "Sample Text") )
            {
                return (MRESULT)FALSE;
            }

            return (MRESULT)hDdf;
        }
    }
    return WinDefWindowProc( hwnd, ulMsg, mp1, mp2 );
}

```

DdfSetColor – Set Color of Text

```
#define INCL_DDF
```

BOOL DdfSetColor (HDDF hddf, COLOR BackColor, COLOR ForColor)

This function sets the background and foreground colors of the displayed text.

Parameters

hddf (HDDF) – input

Handle to DDF returned by DdfInitialize.

BackColor (COLOR) – input

Specifies the desired background color.

ForColor (COLOR) – input

Specifies the desired foreground color.

The following color value constants may be used for the foreground and background colors:

- CLR_DEFAULT - used to set IPF default text color
- CLR_BLACK
- CLR_BLUE
- CLR_RED
- CLR_PINK
- CLR_GREEN
- CLR_CYAN
- CLR_YELLOW
- CLR_BROWN
- CLR_DARKGRAY
- CLR_DARKBLUE
- CLR_DARKRED
- CLR_DARKPINK
- CLR_DARKGREEN
- CLR_DARKCYAN
- CLR_PALEGRAY
- CLR_UNCHANGED

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

HMERR_DDF_MEMORY	Not enough memory is available.
HMERR_DDF_BACKCOLOR	The background color is not valid.
HMERR_DDF_FORCOLOR	The foreground color is not valid.

Remarks

Note: There is a 4-byte ESC code overhead in the DDF internal buffer for the foreground color, and a 4-byte overhead for the background color, with this function.

Related Functions

- DdfText
- DdfSetTextAlign
- DdfSetFormat
- DdfSetFontStyle
- DdfSetFont
- DdfPara
- DdfMetafile
- DdfListItem
- DdfInitialize
- DdfInform
- DdfHyperText
- DdfEndList
- DdfBitmap
- DdfBeginList

Example Code

After initializing a DDF buffer with DdfInitialize, the example uses DdfPara to start a new paragraph, DdfSetFont and DdfSetFontStyle to have the text displayed in a large, bold Courier font, DdfSetColor to change the text color, and DdfText to place text in the buffer. For a more detailed example and discussion of initializing DDF, see the DdfInitialize sample.

```
#define INCL_WINWINDOWMGR      /* General window management */
#define INCL_WINMESSAGEMGR    /* Message management */
#define INCL_DDF               /* Dynamic Data Facility */
#include <os2.h>
#include <pmhelp.h>

MRESULT WindowProc( HWND hwnd, ULONG ulMsg, MPARAM mp1, MPARAM mp2 )
{
    HWND    hwndParent;
    HWND    hwndInstance;    /* help instance window */
    Hddf    hDdf;            /* DDF handle */

    switch( ulMsg )
    {
        case HM_QUERY_DDF_DATA:
            /* get the help instance */
            hwndParent = WinQueryWindow( hwnd, QW_PARENT );
            hwndParent = WinQueryWindow( hwndParent, QW_PARENT );
            hwndInstance = (HWND)WinSendMsg( hwndParent, HM_QUERY,
                                             MPFROMSHORT( HMQW_INSTANCE ), NULL );

            /* Allocate 1K Buffer (default) */
            hDdf = DdfInitialize(
                hwndInstance, /* Handle of help instance */
                0L,          /* Default buffer size */
                0L,          /* Default increment */
            );

            if (hDdf == NULLHANDLE) /* Check return code */
            {
                return (MRESULT)FALSE;
            }
    }
}
```


DdfSetColor — Set Color of Text

```
/* create paragraph in DDF buffer */
if( !DdfPara( hDdf ) )
{
    return (MRESULT)FALSE;
}

/* Change to large (100 x 100 dimensions) Courier font */
if( !DdfSetFont( hDdf, "Courier", 100L, 100L ) )
{
    return (MRESULT)FALSE;
}

/* make the font BOLDFAE */
if( !DdfSetFontStyle( hDdf, FM_SEL_BOLD ) )
{
    return (MRESULT)FALSE;
}

/* make the text display as BLUE on a PALE GRAY background */
if( !DdfSetColor( hDdf, CLR_PALEGRAY, CLR_BLUE ) )
{
    return (MRESULT)FALSE;
}

/* Write data into the buffer */
if ( !DdfText(hDdf, "Sample Text"))
{
    return (MRESULT)FALSE;
}

return (MRESULT)hDdf;
}
return WinDefWindowProc( hwnd, uMsg, mp1, mp2 );
}
```

DdfSetFont – Specify Text Font

```
#define INCL_DDF
```

BOOL DdfSetFont (HDDF hddf, PSZ pszFaceName, ULONG ulWidth, ULONG ulHeight)

This function specifies a text font in the DDF buffer.

Parameters

hddf (HDDF) – input

Handle to DDF returned by DdfInitialize.

pszFaceName (PSZ) – input

This parameter can be specified in two ways:

An ASCIIZ string specifying the font name.

“NULL” or “DEFAULT” to specify the default font.

ulWidth (ULONG) – input

Font width in points. A point is approximately 1/72 of an inch

ulHeight (ULONG) – input

Font height in points.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

HMERR_DDF_MEMORY

Not enough memory is available.

Related Functions

- DdfText
- DdfSetTextAlign
- DdfSetFormat
- DdfSetFontStyle
- DdfSetColor
- DdfPara
- DdfMetafile
- DdfListItem
- DdfInitialize
- DdfInform
- DdfHyperText
- DdfEndList
- DdfBitmap
- DdfBeginList

DdfSetFont — Specify Text Font

Example Code

After initializing a DDF buffer with DdfInitialize, the example uses DdfPara to start a new paragraph, DdfSetFont and DdfSetFontStyle to have the text displayed in a large, bold Courier font, DdfSetColor to change the text color, and DdfText to place text in the buffer. For a more detailed example and discussion of initializing DDF, see the DdfInitialize sample.

```
#define INCL_WINWINDOWMGR      /* General window management */
#define INCL_WINMESSAGEGR     /* Message management */
#define INCL_DDF              /* Dynamic Data Facility */
#include <os2.h>
#include <pmhelp.h>

MRESULT WindowProc( HWND hwnd, ULONG ulMsg, MPARAM mp1, MPARAM mp2 )
{
    HWND    hwndParent;
    HWND    hwndInstance; /* help instance window */
    HDDF    hDdf; /* DDF handle */

    switch( ulMsg )
    {
        case HM_QUERY_DDF_DATA:
            /* get the help instance */
            hwndParent = WinQueryWindow( hwnd, QW_PARENT );
            hwndParent = WinQueryWindow( hwndParent, QW_PARENT );
            hwndInstance = (HWND)WinSendMsg( hwndParent, HM_QUERY,
                                             MPFROMSHORT( HMQW_INSTANCE ), NULL );

            /* Allocate 1K Buffer (default) */
            hDdf = DdfInitialize(
                hwndInstance, /* Handle of help instance */
                0L,          /* Default buffer size */
                0L,          /* Default increment */
            );

            if (hDdf == NULLHANDLE) /* Check return code */
            {
                return (MRESULT)FALSE;
            }

            /* create paragraph in DDF buffer */
            if( !DdfPara( hDdf ) )
            {
                return (MRESULT)FALSE;
            }

            /* Change to large (100 x 100 dimensions) Courier font */
            if( !DdfSetFont( hDdf, "Courier", 100L, 100L ) )
            {
                return (MRESULT)FALSE;
            }

            /* make the font BOLDFACE */
            if( !DdfSetFontStyle( hDdf, FM_SEL_BOLD ) )
            {
                return (MRESULT)FALSE;
            }
    }
}
```

DdfSetFont — Specify Text Font

```
/* make the text display as BLUE on a PALE GRAY background */
if( !DdfSetColor( hDdf, CLR_PALEGRAY, CLR_BLUE ) )
{
    return (MRESULT)FALSE;
}

/* Write data into the buffer */
if ( !DdfText(hDdf, "Sample Text") )
{
    return (MRESULT)FALSE;
}

return (MRESULT)hDdf;
}
return WinDefWindowProc( hwnd, ulMsg, mp1, mp2 );
}
```

DdfSetFontStyle — Specify Text Font Style

```
#define INCL_DDF
```

BOOL DdfSetFontStyle (HDDF hddf, ULONG fFontStyle)

This function specifies a text font style in the DDF buffer.

Parameters

hddf (HDDF) — input

Handle to DDF returned by DdfInitialize.

fFontStyle (ULONG) — input

A NULL value for this parameter will set the font-style back to the default. Any of the following values can be specified:

FM_SEL_ITALIC
FM_SEL_BOLD
FM_SEL_UNDERSCORE

These values can be “ORed” together to combine different font styles.

Note: There is a 4-byte ESC code overhead in the DDF internal buffer for FontStyle.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

HMERR_DDF_MEMORY	Not enough memory is available.
HMERR_DDF_FONTSTYLE	The font style is not valid.

Related Functions

- DdfText
- DdfSetTextAlign
- DdfSetFormat
- DdfSetFont
- DdfSetColor
- DdfPara
- DdfMetafile
- DdfListItem
- DdfInitialize
- DdfInform
- DdfHyperText
- DdfEndList
- DdfBitmap
- DdfBeginList

DdfSetFontStyle — Specify Text Font Style

Example Code

After initializing a DDF buffer with DdfInitialize, the example uses DdfPara to start a new paragraph, DdfSetFont and DdfSetFontStyle to have the text displayed in a large, bold Courier font, DdfSetColor to change the text color, and DdfText to place text in the buffer. For a more detailed example and discussion of initializing DDF, see the DdfInitialize sample.

```
#define INCL_WINWINDOWMGR      /* General window management */
#define INCL_WINMESSAGEMGR    /* Message management */
#define INCL_DDF               /* Dynamic Data Facility */
#include <os2.h>
#include <pmhelp.h>

MRESULT WindowProc( HWND hwnd, ULONG ulMsg, MPARAM mp1, MPARAM mp2 )
{
    HWND    hwndParent;
    HWND    hwndInstance; /* help instance window */
    Hddf    hDdf; /* DDF handle */

    switch( ulMsg )
    {
        case HM_QUERY_DDF_DATA:
            /* get the help instance */
            hwndParent = WinQueryWindow( hwnd, QW_PARENT );
            hwndParent = WinQueryWindow( hwndParent, QW_PARENT );
            hwndInstance = (HWND)WinSendMsg( hwndParent, HM_QUERY,
                                             MPFROMSHORT( HMQW_INSTANCE ), NULL );

            /* Allocate 1K Buffer (default) */
            hDdf = DdfInitialize(
                hwndInstance, /* Handle of help instance */
                0L,           /* Default buffer size */
                0L            /* Default increment */
            );

            if (hDdf == NULLHANDLE) /* Check return code */
            {
                return (MRESULT)FALSE;
            }

            /* create paragraph in DDF buffer */
            if( !DdfPara( hDdf ) )
            {
                return (MRESULT)FALSE;
            }

            /* Change to large (100 x 100 dimensions) Courier font */
            if( !DdfSetFont( hDdf, "Courier", 100L, 100L ) )
            {
                return (MRESULT)FALSE;
            }

            /* make the font BOLDFACE */
            if( !DdfSetFontStyle( hDdf, FM_SEL_BOLD ) )
            {
                return (MRESULT)FALSE;
            }
    }
}
```

DdfSetFontStyle — Specify Text Font Style

```
/* make the text display as BLUE on a PALE GRAY background */
if( IDdfSetColor( hDdf, CLR_PALEGRAY, CLR_BLUE ) )
{
    return (MRESULT)FALSE;
}

/* Write data into the buffer */
if (!DdfText(hDdf, "Sample Text"))
{
    return (MRESULT)FALSE;
}

return (MRESULT)hDdf;
}
return WinDefWindowProc( hwnd, uMsg, mp1, mp2 );
}
```

```
#define INCL_DDF
```

BOOL DdfSetFormat (HDDF hddf, ULONG fFormatType)

This function is used to turn formatting off or on. It corresponds to the **:lines.** tag.

Parameters

hddf (HDDF) – input

Handle to DDF returned by DdfInitialize.

fFormatType (ULONG) – input

Only the following constants may be used in this parameter:

TRUE	Turn formatting on.
FALSE	Turn formatting off.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

HMERR_DDF_MEMORY	Not enough memory is available.
-------------------------	---------------------------------

Remarks

Note: If formatting is ON, there is a 3-byte ESC code overhead in the DDF internal buffer for this function. Otherwise, there is a 4-byte ESC code overhead.

Related Functions

- DdfText
- DdfSetTextAlign
- DdfSetFontStyle
- DdfSetFont
- DdfSetColor
- DdfPara
- DdfMetafile
- DdfListItem
- DdfInitialize
- DdfInform
- DdfHyperText
- DdfEndList
- DdfBitmap
- DdfBeginList

DdfSetFormat — Control Formatting

Example Code

After initializing a DDF buffer with DdfInitialize, the example uses DdfSetTextAlign to specify left justified text in the DDF buffer when formatting is OFF. The example then uses DdfSetFormat to turn off formatting for text in the DDF buffer (corresponds to the IPF lines tag). For a more detailed example and discussion of initializing DDF, see the DdfInitialize sample.

```
#define INCL_WINWINDOWMGR      /* General window management */
#define INCL_WINMESSAGEMGR    /* Message management */
#define INCL_GPIPRIMITIVES    /* Drawing Primitives/Attributes*/
#define INCL_DDF              /* Dynamic Data Facility */
#include <os2.h>
#include <pmhelp.h>

MRESULT WindowProc( HWND hwnd, ULONG ulMsg, MPARAM mp1, MPARAM mp2 )
{
    HWND    hwndParent;
    HWND    hwndInstance;    /* help instance window */
    HDDF    hDdf;            /* DDF handle */

    switch( ulMsg )
    {
        case HM_QUERY_DDF_DATA:
            /* get the help instance */
            hwndParent = WinQueryWindow( hwnd, QW_PARENT );
            hwndParent = WinQueryWindow( hwndParent, QW_PARENT );
            hwndInstance = (HWND)WinSendMsg( hwndParent, HM_QUERY,
                MPFROMSHORT( HMQW_INSTANCE ), NULL );

            /* Allocate 1K Buffer (default) */
            hDdf = DdfInitialize(
                hwndInstance, /* Handle of help instance */
                0L,          /* Default buffer size */
                0L           /* Default increment */
            );

            if (hDdf == NULLHANDLE) /* Check return code */
            {
                return (MRESULT)FALSE;
            }

            /* left justify text when formatting is OFF */
            if (IDdfSetTextAlign(hDdf, TA_LEFT))
            {
                return (MRESULT)FALSE;
            }

            /* turn formatting OFF */
            if (IDdfSetFormat(hDdf, FALSE))
            {
                return (MRESULT)FALSE;
            }

            if (IDdfText(hDdf,
                "Format OFF: This text should be Left Aligned!\n"))
            {
                return (MRESULT)FALSE;
            }

            return (MRESULT)hDdf;
        }
    }
    return WinDefWindowProc( hwnd, ulMsg, mp1, mp2 );
}
```

DdfSetTextAlign – Define Text Alignment

```
#define INCL_DDF
```

BOOL DdfSetTextAlign (HDDF hddf, ULONG fAlign)

This function defines whether left, center, or right text justification is to be used when text formatting is off.

Parameters

hddf (HDDF) – input

Handle to DDF returned by DdfInitialize.

fAlign (ULONG) – input

Only the following constants may be used:

TA_LEFT	Left-justify text.
TA_RIGHT	Right-justify text.
TA_CENTER	Center text.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

HMERR_DDF_ALIGN_TYPE	The alignment type is not valid.
-----------------------------	----------------------------------

Remarks

It should be called before DdfSetFormat is called to turn off text formatting, and should not be called again until formatting is turned back on. Note that leading and trailing spaces are not stripped from the text as a result of this alignment.

Related Functions

- DdfText
- DdfSetFormat
- DdfSetFontStyle
- DdfSetFont
- DdfSetColor
- DdfPara
- DdfMetafile
- DdfListItem
- DdfInitialize
- DdfInform
- DdfHyperText
- DdfEndList
- DdfBitmap
- DdfBeginList

DdfSetTextAlign — Define Text Alignment

Example Code

After initializing a DDF buffer with DdfInitialize, the example uses DdfSetTextAlign to specify left justified text in the DDF buffer when formatting is OFF. The example then uses DdfSetFormat to turn off formatting for text in the DDF buffer (corresponds to the IPF lines tag). For a more detailed example and discussion of initializing DDF, see the DdfInitialize sample.

```
#define INCL_WINWINDOWMGR      /* General window management */
#define INCL_WINMESSAGEMGR    /* Message management */
#define INCL_GPIPRIMITIVES    /* Drawing Primitives/Attributes*/
#define INCL_DDF              /* Dynamic Data Facility */
#include <os2.h>
#include <pmhelp.h>

MRESULT WindowProc( HWND hwnd, ULONG ulMsg, MPARAM mp1, MPARAM mp2 )
{
    HWND  hwndParent;
    HWND  hwndInstance;    /* help instance window */
    Hddf  hDdf;            /* DDF handle */

    switch( ulMsg )
    {
        case HM_QUERY_DDF_DATA:
            /* get the help instance */
            hwndParent = WinQueryWindow( hwnd, QW_PARENT );
            hwndParent = WinQueryWindow( hwndParent, QW_PARENT );
            hwndInstance = (HWND)WinSendMsg( hwndParent, HM_QUERY,
                                             MPFROMSHORT( HMQW_INSTANCE ), NULL );

            /* Allocate 1K Buffer (default) */
            hDdf = DdfInitialize(
                hwndInstance, /* Handle of help instance */
                0L,          /* Default buffer size */
                0L           /* Default increment */
            );

            if (hDdf == NULLHANDLE) /* Check return code */
            {
                return (MRESULT)FALSE;
            }

            /* left justify text when formatting is OFF */
            if (!DdfSetTextAlign(hDdf, TA_LEFT))
            {
                return (MRESULT)FALSE;
            }

            /* turn formatting OFF */
            if (!DdfSetFormat(hDdf, FALSE))
            {
                return (MRESULT)FALSE;
            }

            if (!DdfText(hDdf,
                "Format OFF: This text should be Left Aligned!\n"))
            {
                return (MRESULT)FALSE;
            }

            return (MRESULT)hDdf;
        }
    }
    return WinDefWindowProc( hwnd, ulMsg, mp1, mp2 );
}
```

DdfText – Add Text to DDF Buffer

```
#define INCL_DDF
```

BOOL DdfText (HDDF hddf, PSZ pszText)

This function adds text to the DDF buffer.

Parameters

hddf (HDDF) – input

Handle to DDF returned by DdfInitialize.

pszText (PSZ) – input

Pointer to the text buffer to be formatted.

Note: There is a 3-byte ESC code overhead in the DDF internal buffer for each word in the text buffer. There is a 1-byte ESC code overhead for each blank and for each newline character.

Returns

Success indicator.

TRUE Successful completion.

FALSE Error occurred.

Related Functions

- DdfSetTextAlign
- DdfSetFormat
- DdfSetFontStyle
- DdfSetFont
- DdfSetColor
- DdfPara
- DdfMetafile
- DdfListItem
- DdfInitialize
- DdfInform
- DdfHyperText
- DdfEndList
- DdfBitmap
- DdfBeginList

Example Code

After initializing a DDF buffer with DdfInitialize, the example uses DdfPara to start a new paragraph, DdfSetFont and DdfSetFontStyle to have the text displayed in a large, bold Courier font, DdfSetColor to change the text color, and DdfText to place text in the buffer. For a more detailed example and discussion of initializing DDF, see the DdfInitialize sample.

```
#define INCL_WINWINDOWMGR    /* General window management */
#define INCL_WINMESSAGEMGR   /* Message management */
#define INCL_DDF              /* Dynamic Data Facility */
#include <os2.h>
#include <pmhelp.h>
```

```
MRESULT WindowProc( HWND hwnd, ULONG ulMsg, MPARAM mp1, MPARAM mp2 )
{
```

DdfText —

Add Text to DDF Buffer

```
HWND    hwndParent;
HWND    hwndInstance;    /* help instance window    */
HDDF    hDdf;            /* DDF handle    */

switch( uMsg )
{
case HM_QUERY_DDF_DATA:
    /* get the help instance */
    hwndParent = WinQueryWindow( hwnd, QW_PARENT );
    hwndParent = WinQueryWindow( hwndParent, QW_PARENT );
    hwndInstance = (HWND)WinSendMsg( hwndParent, HM_QUERY,
                                    MPFROMSHORT( HMQW_INSTANCE ), NULL );

    /* Allocate 1K Buffer (default) */
    hDdf = DdfInitialize(
        hwndInstance, /* Handle of help instance */
        0L,           /* Default buffer size    */
        0L,           /* Default increment      */
    );

    if (hDdf == NULLHANDLE)    /* Check return code    */
    {
        return (MRESULT)FALSE;
    }

    /* create paragraph in DDF buffer */
    if( !DdfPara( hDdf ) )
    {
        return (MRESULT)FALSE;
    }

    /* Change to large (100 x 100 dimensions) Courier font */
    if( !DdfSetFont( hDdf, "Courier", 100L, 100L ) )
    {
        return (MRESULT)FALSE;
    }

    /* make the font BOLDFACE */
    if( !DdfSetFontStyle( hDdf, FM_SEL_BOLD ) )
    {
        return (MRESULT)FALSE;
    }

    /* make the text display as BLUE on a PALE GRAY background */
    if( !DdfSetColor( hDdf, CLR_PALEGRAY, CLR_BLUE ) )
    {
        return (MRESULT)FALSE;
    }

    /* Write data into the buffer */
    if ( !DdfText(hDdf, "Sample Text") )
    {
        return (MRESULT)FALSE;
    }

    return (MRESULT)hDdf;
}
return WinDefWindowProc( hwnd, uMsg, mp1, mp2 );
}
```

Chapter 5. Graphics Functions

Coordinates

GPI coordinate values that are in world or model space are passed in variables of data type LONG. For a presentation space of format GPIF_LONG (see GpiCreatePS), the signed value must be contained within the low-order 28 bits.

For a presentation space with a format of GPIF_SHORT, the signed value must be contained within the low-order 16 bits. Coordinates that exceed this limit are truncated without error, when stored in a segment. As a consequence, a large positive number may appear as a negative number.

In both instances, after transformation to media space (that is, device space, possibly including a translation for the window origin), coordinate values must be in the range $-32\,768$ through $+32\,767$.

The PMERR_COORDINATE_OVERFLOW error condition occurs if a coordinate is too large to be handled.

Region coordinates must be within the range $-32\,767$ through $+32\,765$.

Matrix Parameter Values

These GPI functions define transforms:

- GpiSetSegmentTransformMatrix
- GpiSetModelTransformMatrix
- GpiCallSegmentMatrix
- GpiSetViewingTransformMatrix
- GpiSetDefaultViewMatrix
- GpiCreatePS
- GpiSetPageViewport.

Note: The last two functions define the device transform; the page viewport may be defaulted.

Concatenation of transform matrixes can occur as the transform is specified, for example, if TRANSFORM_ADD is specified. Concatenation also occurs during drawing, between the various transforms in the viewing pipeline.

During the process of concatenation, it is possible for the matrix parameter overflow error, PMERR_INV_MATRIX_ELEMENT, to occur. This error is raised if either of the following conditions occurs for any intermediate value during the concatenation arithmetic (see, for example, GpiSetSegmentTransformMatrix for an explanation of matrix element numbers):

- Any of the matrix elements 1, 2, 4, or 5 is greater than $32\,767$ or less than $-32\,768$ (± 1 for a GPIF_SHORT format presentation space), or
- Either of elements 7 or 8 is greater than $134\,217\,727$ ($2^{27} - 1$) or less than $-134\,217\,728$ (-2^{27}) (greater than $32\,767$ or less than $-32\,768$ for a GPIF_SHORT format presentation space).

Rounding Errors

In general for graphics coordinates, when non-unity transforms (apart from simple translation) are involved, rounding errors occur. For example, adding the coordinates of one point to a delta value, to produce the coordinates of a second point (all in world coordinates) does not always map to the same device pel as if the computation had been done in device coordinates. Such errors can be avoided if calculations are done in device coordinates, or if there are no scaling (or rotational, or shear) elements in the transforms. Alternatively, the problems can be reduced, though not eliminated, by defining very fine world coordinates.

Drawing Process Check Errors

Some GPI functions involve processing buffers of graphics orders or retained graphics segments (the data for which consists of graphics orders). These functions can give rise to Drawing Process Check (DPC) errors if an order is found that either is not valid in its context or that contains invalid data. If this happens, processing of the function stops and the error is recorded. Note that orders up to the one found to be in error are processed by the function, and output occurs if drawing is being performed.

Each function that can return these errors has Drawing Process Check errors in its error condition list. The full list of DPC errors is:

- PMERR_INV_IN_AREA
- PMERR_INV_IN_PATH
- PMERR_INV_IN_ELEMENT
- PMERR_ALREADY_IN_ELEMENT
- PMERR_STOP_DRAW_OCCURRED (warning)
- PMERR_PATH_INCOMPLETE
- PMERR_AREA_INCOMPLETE
- PMERR_IMAGE_INCOMPLETE
- PMERR_INV_ORDER_LENGTH
- PMERR_NOT_IN_IMAGE
- PMERR_NOT_IN_AREA
- PMERR_NOT_IN_ELEMENT
- PMERR_NOT_IN_PATH
- PMERR_INSUFFICIENT_MEMORY
- PMERR_SEG_CALL_STACK_EMPTY
- PMERR_SEG_CALL_STACK_FULL
- PMERR_TRUNCATED_ORDER
- PMERR_CALLED_SEG_NOT_FOUND
- PMERR_DYNAMIC_SEG_SEQ_ERROR
- PMERR_PROLOG_ERROR
- PMERR_INV_IN_VECTOR_SYMBOL

GPI Functions by Functional Area

The following table shows how all of the Graphics Programming Interface (GPI) functions are related within functional areas.

C Name	C Name
Curve Functions	
Attribute Setting Functions	
GpiQueryArcParams	GpiSetArcParams
GpiQueryDefArcParams	GpiSetDefArcParams
Primitive Functions	
GpiFullArc	GpiPolyFillet
GpiPartialArc	GpiPolyFilletSharp
GpiPointArc	GpiPolySpline
Area Functions	
Attribute Setting Functions	
GpiQueryPattern	GpiSetPattern
GpiQueryPatternRefPoint	GpiSetPatternRefPoint
GpiQueryPatternSet	GpiSetPatternSet

C Name	C Name
Primitive Functions	
GpiBeginArea	GpiEndArea
Bit-Map Support	
Creation and Selection Functions	
GpiCreateBitmap	GpiQueryBitmapDimension
GpiDeleteBitmap	GpiSetBitmap
GpiLoadBitmap	GpiSetBitmapDimension
Operations on Raw Bit Maps	
GpiQueryBitmapBits	GpiQueryDeviceBitmapFormats
GpiQueryBitmapInfoHeader	GpiSetBitmapBits
GpiQueryBitmapParameters	
Operations through Presentation Spaces	
GpiBitBlt	GpiSetPel
GpiDrawBits	GpiWCBitBlt
GpiQueryPel	
Resources and Defaults Functions	
GpiQueryBitmapHandle	GpiSetBitmapId
Character Functions	
Attribute Setting Functions	
GpiQueryCharAngle	GpiSetCharAngle
GpiQueryCharBox	GpiSetCharBox
GpiQueryCharBreakExtra	GpiSetCharBreakExtra
GpiQueryCharDirection	GpiSetCharDirection
GpiQueryCharExtra	GpiSetCharExtra
GpiQueryCharMode	GpiSetCharMode
GpiQueryCharSet	GpiSetCharSet
GpiQueryCharShear	GpiSetCharShear
GpiQueryTextAlignment	GpiSetTextAlignment
Primitive Functions	
GpiCharString	GpiCharStringPosAt
GpiCharStringAt	GpiQueryCharStringPos
GpiCharStringPos	GpiQueryCharStringPosAt
Resources and Defaults Functions	
GpiCreateLogFont	GpiQueryKerningPairs
GpiDeleteSetId	GpiQueryLogicalFont
GpiLoadFonts	GpiQueryNumberSetIds
GpiLoadPublicFonts	GpiQuerySetIds
GpiQueryCp	GpiQueryTextBox
GpiQueryDefCharBox	GpiQueryWidthTable
GpiQueryFaceString	GpiSetCp

C Name	C Name
GpiQueryFontMetrics	GpiUnloadFonts
GpiQueryFonts	GpiUnloadPublicFonts
GpiQueryFullFontFileDescriptions	GpiQueryFontAction
Color and Mix Functions	
Attribute Setting Functions	
GpiQueryBackColor	GpiSetBackColor
GpiQueryBackMix	GpiSetBackMix
GpiQueryColor	GpiSetColor
GpiQueryMix	GpiSetMix
Resources and Default Functions	
GpiCreateLogColorTable	GpiQueryNearestColor
GpiQueryColorData	GpiQueryRealColors
GpiQueryColorIndex	GpiQueryRGBColor
GpiQueryLogColorTable	
Palette Manager Functions	
GpiAnimatePalette	GpiQueryPaletteInfo
GpiCreatePalette	GpiSelectPalette
GpiDeletePalette	GpiSetPaletteEntries
GpiQueryPalette	
Control Functions	
GpiAssociate	GpiQueryPS
GpiCreatePS	GpiResetPS
GpiDestroyPS	GpiRestorePS
GpiErrorSegmentData	GpiSavePS
GpiQueryDevice	GpiSetPS
Correlation and Boundary Determination Functions	
Bounds Data Functions	
GpiQueryBoundaryData	GpiResetBoundaryData
Correlation Data Functions	
GpiCorrelateChain	GpiCorrelateSegment
GpiCorrelateFrom	
Pick Aperture and Tag Functions	
GpiQueryDefTag	GpiSetDefTag
GpiQueryPickAperturePosition	GpiSetPickAperturePosition
GpiQueryPickApertureSize	GpiSetPickApertureSize
GpiQueryTag	GpiSetTag
Drawing Functions	
GpiDrawChain	GpiQueryDrawControl
GpiDrawDynamics	GpiQueryDrawingMode
GpiDrawFrom	GpiQueryStopDraw

C Name	C Name
GpiDrawSegment	GpiRemoveDynamics
GpiErase	GpiSetDrawControl
GpiFloodFill	GpiSetDrawingMode
GpiGetData	GpiSetStopDraw
GpiPutData	GpiPolygons
General Attribute Functions	
Attribute Mode Functions	
GpiPop	GpiSetAttrMode
GpiQueryAttrMode	GpiSetDefAttrs
GpiQueryDefAttrs	
Attribute Strip Setting Functions	
GpiQueryAttrs	GpiSetAttrs
Image Functions	
Primitive Functions	
GpiImage	
Line Functions	
Attribute Setting Functions	
GpiQueryLineEnd	GpiSetLineEnd
GpiQueryLineJoin	GpiSetLineJoin
GpiQueryLineType	GpiSetLineType
GpiQueryLineWidth	GpiSetLineWidth
GpiQueryLineWidthGeom	GpiSetLineWidthGeom
Primitive Functions	
GpiBox	GpiPolyLine
GpiLine	GpiQueryCurrentPosition
GpiMove	GpiSetCurrentPosition
GpiPolyLineDisjoint	
Visibility Functions	
GpiPtVisible	GpiRectVisible
Marker Functions	
Attribute Setting Functions	
GpiQueryMarker	GpiSetMarker
GpiQueryMarkerBox	GpiSetMarkerBox
GpiQueryMarkerSet	GpiSetMarkerSet
Primitive Functions	
GpiMarker	GpiPolyMarker
Metafile Support	
GpiCopyMetaFile	GpiQueryMetaFileBits
GpiDeleteMetaFile	GpiQueryMetaFileLength
GpiLoadMetaFile	GpiSaveMetaFile

C Name	C Name
GpiPlayMetaFile	GpiSetMetaFileBits
Miscellaneous Functions	
GpiComment	
Path Functions	
Path Clipping Functions	
GpiSetClipPath	
Path Definition and Deletion Functions	
GpiBeginPath	GpiEndPath
GpiCloseFigure	
Path Drawing Functions	
GpiFillPath	GpiStrokePath
GpiOutlinePath	
Path Manipulation Functions	
GpiModifyPath	
Region Support	
Clipping Region Functions	
GpiExcludeClipRectangle	GpiQueryClipBox
GpiIntersectClipRectangle	GpiQueryClipRegion
GpiOffsetClipRegion	GpiSetClipRegion
Drawing Functions	
GpiFrameRegion	GpiPaintRegion
Region Functions	
GpiCombineRegion	GpiPtInRegion
GpiCreateRegion	GpiQueryRegionBox
GpiDestroyRegion	GpiQueryRegionRects
GpiEqualRegion	GpiRectInRegion
GpiOffsetRegion	GpiSetRegion
GpiPathToRegion	
Segment Manipulation Functions	
Segment Content Manipulation Functions	
GpiBeginElement	GpiQueryEditMode
GpiDeleteElement	GpiQueryElement
GpiDeleteElementRange	GpiQueryElementPointer
GpiDeleteElementsBetweenLabels	GpiQueryElementType
GpiElement	GpiSetEditMode
GpiEndElement	GpiSetElementPointer
GpiLabel	GpiSetElementPointerAtLabel
GpiOffsetElementPointer	
Whole Segment Functions	
GpiCloseSegment	GpiQuerySegmentNames

C Name	C Name
GpiDeleteSegment	GpiQuerySegmentPriority
GpiDeleteSegments	GpiSetInitialSegmentAttrs
GpiOpenSegment	GpiSetSegmentAttrs
GpiQueryInitialSegmentAttrs	GpiSetSegmentPriority
GpiQuerySegmentAttrs	
Transform Functions	
Clipping	
GpiQueryDefViewingLimits	GpiSetDefViewing Limits
GpiQueryGraphicsField	GpiSetGraphicsField
GpiQueryViewingLimits	GpiSetViewingLimits
Conversion Functions	
GpiConvert	GpiConvertWithMatrix
Device Transforms	
GpiQueryPageViewport	GpiSetPageViewport
Helper Functions	
GpiRotate	GpiTranslate
GpiScale	
Modelling Transform Functions	
GpiCallSegmentMatrix	GpiSetModelTransformMatrix
GpiQueryModelTransformMatrix	GpiSetSegmentTransformMatrix
GpiQuerySegmentTransformMatrix	
Viewing Transform Functions	
GpiQueryDefaultViewMatrix	GpiSetDefaultViewMatrix
GpiQueryViewingTransformMatrix	GpiSetViewingTransformMatrix

GpiAnimatePalette — Animate Palette

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI or INCL_PM */
```

LONG GpiAnimatePalette (HPAL hpal, ULONG ulFormat, ULONG ulStart, ULONG ulCount, PULONG aulTable)

This function changes the color values of animating indexes in a palette.

Parameters

hpal (HPAL) — input
Palette handle.

ulFormat (ULONG) — input
Format of entries in the table:

LCOLF_CONSECRGB Array of RGB values, corresponding to color indexes *ulStart* upwards.
Each entry is 4 bytes long.

ulStart (ULONG) — input
Starting index.

This is relevant only for LCOLF_CONSECRGB.

ulCount (ULONG) — input
Count of elements in *aulTable*.

This must be greater than or equal to 0.

aulTable (PULONG) — input
Start of the application data area.

This contains the palette definition data. The format depends on the value of *ulFormat*.

Each color value is a 4-byte integer, with a value of

$(F * 16777216) + (R * 65536) + (G * 256) + B$

where:

F is a flag byte, which can take the following values (these can be ORed together if required):

PC_RESERVED This index is an animating index. This means that the application might frequently change the RGB value, so the system should not map the logical index of the palette of another application to the entry in the physical palette used for this color.

PC_EXPLICIT The low-order word of the logical color table entry designates a physical palette entry. This allows an application to show the contents of the device palette as realized for other logical palettes. This does not prevent the color in the entry from being changed for any reason.

R is red intensity value

G is green intensity value

B is blue intensity value.

The maximum intensity for each primary is 255.

GpiAnimatePalette – Animate Palette

Returns

Number of remapped colors.

PAL_ERROR Error occurred

Other Number of colors remapped (that is, having entries in the physical color table). These are all animating indexes: they have the PC_RESERVED flag set on this function. If the palette is selected into more than one presentation space, the number returned is the maximum number of indexes that have entries in any of the relevant devices.

Note that by the time an application receives this information, other applications using the palette may have caused the number to be changed.

Possible returns from WinGetLastError

PMERR_INV_HPAL	An invalid color palette handle was specified.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_INV_COLOR_DATA	Invalid color table definition data was specified with GpiCreateLogColorTable.
PMERR_INV_COLOR_FORMAT	An invalid format parameter was specified with GpiCreateLogColorTable.
PMERR_INV_COLOR_START_INDEX	An invalid starting index parameter was specified with a logical color table or color query function.
PMERR_INSUFFICIENT_MEMORY	The operation terminated through insufficient memory.
PMERR_PALETTE_BUSY	An attempt has been made to reset the owner of a palette when it was busy.
PMERR_INV_IN_AREA	An attempt was made to issue a function invalid inside an area bracket. This can be detected while the actual drawing mode is draw or draw-and-retain or during segment drawing or correlation functions.

Remarks

The animating indexes are those that have the PC_RESERVED flag set in the palette and also in the corresponding element of the *auTable* array in this function.

If an animating index already has an entry in the physical hardware palette (allocated from a previous call to WinRealizePalette), both that entry and the entry in the logical palette are changed. If there is not an entry in the physical palette, or the device does not support palette functions, the logical palette color is changed. This function does not allocate a new entry in the physical palette.

This function ignores those elements in *auTable* corresponding to non-animating indexes (those that do not have the PC_RESERVED flag set). Their colors are not changed.

All presentation spaces that have this palette selected into them (see GpiSelectPalette) are updated with the effects of this function. It is not necessary to issue a WinRealizePalette function before the effects become visible.

If a palette is selected into a presentation space that is associated with a device context of type OD_METAFILE or OD_METAFILE_NOQUERY, only the final color values are recorded in the metafile.

It is an error if a palette is selected into a presentation space that is within an area or path definition when this function is issued.

GpiAnimatePalette —

Animate Palette

Related Functions

- GpiCreatePalette
- GpiDeletePalette
- GpiQueryPalette
- GpiQueryPaletteInfo
- GpiSelectPalette
- GpiSetPaletteEntries
- WinRealizePalette

Example Code

This example uses GpiAnimatePalette to change the color values of the first four animating indexes in a palette.

```
#define INCL_GPILOGCOLORTABLE  /* Color Table functions      */
#include <os2.h>

LONG lremapColors;    /* number of remapped colors */
HPAL hpal;            /* palette handle             */

/*****
 * assume 4 entries in palette.
 * The RGB values are calculated with the following formula:
 *   (F * 16777216) + (R * 65536) + (G * 256) + B
 *   where F = flag, PC_RESERVED or PC_EXPLICIT
 *   R = red intensity value
 *   G = green intensity value
 *   B = blue intensity value
 * Thus, in the following table, red and green intensities are 0
 * while the blue intensity increases from 1 to 4.
 *****/

ULONG auTable[4]=
    {(PC_RESERVED*16777216) + (0*65536) + (0*256) + 1,
     (PC_RESERVED*16777216) + (0*65536) + (0*256) + 2,
     (PC_RESERVED*16777216) + (0*65536) + (0*256) + 3,
     (PC_RESERVED*16777216) + (0*65536) + (0*256) + 4};

lremapColors = GpiAnimatePalette(hpal, LCOLF_CONSECRGB, 0L, 4L,
                                auTable);
```

```
#define INCL_GPICONTROL /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

```
BOOL GpiAssociate (HPS hps, HDC hdc)
```

This function associates a graphics presentation space with, or dissociates it from, a device context.

Parameters

hps (HPS) – input
Presentation-space handle.

hdc (HDC) – input
Device-context handle.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_PS_IS_ASSOCIATED	An attempt was made to destroy a presentation or associate a presentation space that is still associated with a device context.
PMERR_DC_IS_ASSOCIATED	An attempt was made to associate a presentation space with a device context that was already associated or to destroy a device context that was associated.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.
PMERR_INV_HDC	An invalid device-context handle or (micro presentation space) presentation-space handle was specified.
PMERR_REALIZE_NOT_SUPPORTED	An attempt was made to create a realizable logical color table on a device driver that does not support this function.
PMERR_PATH_INCOMPLETE	An attempt was made to open or close a segment either directly or during segment drawing, or to issue GpiAssociate while there is an open path bracket.
PMERR_AREA_INCOMPLETE	Either: <ul style="list-style-type: none"> • A segment has been opened, closed, or drawn. • GpiAssociate was issued while an area bracket was open. • A drawn segment has opened an area bracket and ended without closing it.

GpiAssociate — Associate

Remarks

Any type of device context may be used.

Subsequent drawing functions direct output to the associated device context.

If a null handle is supplied for the device context, the presentation space is dissociated from its currently-associated device context. An associated presentation space cannot be associated with another device context, and an associated device context cannot be associated with another presentation space.

An error occurs if you try to draw to a presentation space associated with a memory device context that has no bit map selected into it (see GpiSetBitmap).

The processing described for GRES_ATTRS (see GpiResetPS) is performed on the presentation space. Also, bounds data is destroyed, the page viewport is reset to its default value (see GpiCreatePS), and any clip region and path definition are lost. The save/restore presentation-space stack (see GpiSavePS) is purged.

Any palette selected into the presentation space remains selected.

Any dynamic segments left drawn on the device are not subsequently removed by GpiRemoveDynamics.

Related Functions

- GpiCreatePS
- GpiDestroyPS
- GpiQueryDevice
- GpiQueryPS
- GpiResetPS
- GpiRestorePS
- GpiSavePS
- GpiSetPS
- GpiSetMarkerSet
- GpiSetPatternSet

Example Code

This example releases the current device context and associates a new device context with the presentation space.

```
#define INCL_GPICONTROL          /* GPI control Functions      */
#include <os2.h>

HPS hps;                        /* presentation space handle */
HDC hdcPrinter;                 /* device context handle      */

/* release the current device context */
GpiAssociate(hps, NULLHANDLE);
/* associate a printer device context */
GpiAssociate(hps, hdcPrinter);
```

GpiBeginArea – Begin Area

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

BOOL GpiBeginArea (HPS hps, ULONG fIOptions)

This function begins the construction of an area.

Parameters

hps (HPS) – input
Presentation-space handle.

fIOptions (ULONG) – input
Area options.

This contains fields of option bits. For each field, one value should be selected (unless the default is suitable). These values can be ORed together to determine whether to draw boundary lines as well as the area interior:

BA_NOBOUNDARY Do not draw boundary lines.

BA_BOUNDARY Draw boundary lines (the default).

Construction of the area interior:

BA_ALTERNATE Construct interior in *alternate* mode (the default)

BA_WINDING Construct interior in *winding* mode.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from GetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_AREA_CONTROL An invalid options parameter was specified with GpiBeginArea.

PMERR_INV_IN_PATH An attempt was made to issue a function invalid inside a path bracket.

PMERR_ALREADY_IN_AREA An attempt was made to begin a new area while an existing area bracket was already open.

Remarks

The construction is terminated by the GpiEndArea function.

You can use the following list of functions to define an area. They are used between the GpiBeginArea and GpiEndArea functions.

- GpiBeginElement
- GpiBox (with the *IControl* parameter set to DRO_OUTLINE)
- GpiCallSegmentMatrix
- GpiComment
- GpiElement (containing a valid call)

GpiBeginArea — Begin Area

- GpiEndElement
- GpiFullArc (with the *IControl* parameter set to DRO_OUTLINE)
- GpiLabel
- GpiLine
- GpiMove
- GpiPartialArc
- GpiPointArc
- GpiPolyFillet
- GpiPolyFilletSharp
- GpiPolyLine
- GpiPolySpline
- GpiPop (that pops a valid call)
- GpiSetArcParams
- GpiSetAttrMode
- GpiSetAttrs (setting valid line attributes only, or foreground color/mix (only) for other primitive types)
- GpiSetColor
- GpiSetCurrentPosition
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetMix
- GpiSetModelTransformMatrix

GpiBox and GpiFullArc are valid only in an area bracket (that is, between the GpiBeginArea and GpiEndArea functions with the *IControl* parameter set to DRO_OUTLINE. Other values of this parameter on these functions cause an implicit area bracket around the function.

Shading of the area is performed using the current pattern, as set by the GpiSetPattern function. The color and color-mixing modes that are current at the time GpiBeginArea is issued define the attributes to be applied to the pattern. The pattern reference point is also subjected to all of the transformations (including the model transformation) in force at the time of GpiBeginArea.

The area boundary consists of one or more *closed* figures, each constructed by:

- GpiBox
- GpiFullArc
- GpiPointArc
- GpiLine
- GpiPartialArc
- GpiPolyFilletSharp
- GpiPolyLine
- GpiPolySpline
- GpiPolyFillet

The GpiSetColor and GpiSetMix functions can be used to control how the area boundary is to be colored. The GpiSetLineEnd, GpiSetLineJoin, GpiSetLineType, and GpiSetLineWidth functions can be used to control line attributes as required. GpiSetAttrs can be used as an alternative way of setting these attributes. GpiSetArcParams can be used to control the shape of arcs produced by GpiFullArc, GpiPointArc, and GpiPartialArc.

The start of a new figure is indicated by:

- GpiCallSegmentMatrix
- GpiFullArc
- GpiMove
- GpiPop (or end of called segment), which pops current position or a model transform
- GpiSetCurrentPosition
- GpiSetModelTransformMatrix

Note: GpiCloseFigure must not be issued within an area.

GpiBeginArea — Begin Area

A GpiBox or GpiFullArc function called within an area definition generates a complete closed figure. These functions must not be used within another figure definition.

The starting point of each closed figure is the current position when this function is made, or the point specified by the function starting the figure. Figure construction continues until either a new figure is started, or GpiEndArea is called.

Each figure should be closed, that is, the start and end points should be identical. If these points are not identical, they are joined by a straight line to arbitrarily close the figure.

The area interior is constructed either in *alternate* mode or in *winding* mode. In alternate mode, whether any point is within the interior is determined by drawing an imaginary line from that point to infinity; if there is an odd number of boundary crossings, the point is inside the area, if there is an even number of crossings, it is not.

In winding mode, the direction of the boundary lines is taken into account. Using the same imaginary line, the number of crossings is counted, as in alternate mode, but boundary lines going in one direction score plus one, and boundary lines going in the other direction score minus one. The point is in the interior if the final score is not zero.

In either mode, all of the boundaries of the area are considered to be part of the interior.

If the *flOptions* parameter of this function is BA_NOBOUNDARY, the boundary lines are *not* drawn, but the shading ends at the boundaries. If the *flOptions* parameter specifies BA_BOUNDARY, the boundary lines and any lines added to close the figures are drawn. The lines are drawn using the current line attributes (which can be changed during construction) and shading occurs within the boundaries.

The current position is not changed by this function, but it can be changed by the moves, arcs, fillets, and lines between this function and the GpiEndArea function, including any used to close figures.

Area definitions cannot be nested. This function and the GpiEndArea function for one area must be within the same segment.

You can have no more than 1 450 straight-line vertices that describe the area.

During correlation in nonretained mode, a hit on any function within an area returns GPI_HITS in the GpiEndArea function. GPI_HITS is not returned on any of the primitives that occur within the area definition.

Related Functions

- GpiBeginPath
- GpiEndArea
- GpiSetPattern
- GpiSetPatternRefPoint
- GpiSetPatternSet
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

GpiBeginArea – Begin Area

Graphic Elements and Orders

Element Type: OCODE_GBAR

Order: Begin Area

Example Code

This example uses the GpiBeginArea function to draw an area. The area, an isosceles triangle, is drawn with boundary lines and filled using the alternate filling mode.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions */
#include <os2.h>

HPS hps;                        /* presentation space handle */
POINTL ptlStart = { 0, 0 }; /* first vertex */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices */

GpiMove(hps, &ptlStart); /* move to starting point (0, 0) */
GpiBeginArea(hps, /* start the area bracket */
    BA_BOUNDARY | /* draw boundary lines */
    BA_ALTERNATE); /* fill interior with alternate mode */
GpiPolyLine(hps, 3L, ptlTriangle); /* draw the triangle */
GpiEndArea(hps); /* end the area bracket */
```

GpiBeginElement – Begin Element

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiBeginElement (HPS hps, LONG IType, PSZ pszDesc)

This function defines the start of an element within a segment.

Parameters

hps (HPS) – input

Presentation-space handle.

IType (LONG) – input

Type to be associated with the element.

Application-defined elements should have type values in the range X'81xxxxxx' through X'FFxxxxxx' to avoid conflict with system-generated elements.

pszDesc (PSZ) – input

Description.

Variable-length character string, recorded with the type.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_ALREADY_IN_ELEMENT

An attempt was made to begin a new element while an existing element bracket was already open.

PMERR_DESC_STRING_TRUNCATED

An attempt was made to supply a description string with GpiBeginElement that was greater than the permitted maximum length (251 characters). The string was truncated.

Remarks

This function starts an element, stored in the current segment, in **retain** or **draw-and-retain** mode (see GpiSetDrawingMode). The element is drawn in **draw** or **draw-and-retain** mode.

The drawing functions that form the contents of the element are passed on subsequent GPI functions (only those functions that can generate orders are logically part of the element). The element extends up to the next GpiEndElement function (or GpiCloseSegment, which causes an implicit GpiEndElement to be generated).

Grouping drawing functions together into an element is useful if the set of functions is to be changed or replaced together at a later time. Drawing functions that are not explicitly grouped together in an element bracket (GpiBeginElement-GpiEndElement pair) generate a single element for each GPI function.

GpiBeginElement — Begin Element

The GpiElement function, that itself generates a complete element, is not allowed within an element bracket. The GpiLabel function is also not allowed within an element bracket. Elements must not be nested within one segment.

Related Functions

- GpiCloseSegment
- GpiDeleteElement
- GpiDeleteElementRange
- GpiDeleteElementsBetweenLabels
- GpiElement
- GpiEndElement
- GpiLabel
- GpiOffsetElementPointer
- GpiQueryElement
- GpiQueryElementPointer
- GpiQueryElementType
- GpiSetElementPointer
- GpiSetElementPointerAtLabel

Graphic Elements and Orders

The element type is defined by the *IType* parameter.

Order: **Begin Element**

Example Code

This example uses the GpiBeginElement function to create an element in a segment. The element type is 1 and the element description is "Triangle". The application can use these later to identify the element.

```
#define INCL_GPISEGEDITING      /* GPI Segment Edit functions */
#include <os2.h>

HPS hps;
POINTL ptlStart = { 0, 0 }; /* first vertex */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices */

GpiBeginElement(hps,          /* start element bracket */
                1L,           /* element type is 1 */
                "Triangle");  /* element description */
GpiMove(hps, &ptlStart);     /* move to start point (0, 0) */
GpiPolyLine(hps, 3L, ptlTriangle); /* draw triangle */
GpiEndElement(hps);          /* end element bracket */
```

GpiBeginPath — Begin Path

```
#define INCL_GPIPATHS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiBeginPath (HPS hps, LONG IPath)

This function specifies the start of a path.

Parameters

hps (HPS) — input
Presentation-space handle.

IPath (LONG) — input
Path identifier.

This must be 1.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PATH_ID

An invalid path identifier parameter was specified.

PMERR_ALREADY_IN_PATH

An attempt was made to begin a new path while an existing path bracket was already open.

PMERR_INV_IN_AREA

An attempt was made to issue a function invalid inside an area bracket. This can be detected while the actual drawing mode is **draw** or **draw-and-retain** or during segment drawing or correlation functions.

Remarks

Paths can be used for these purposes:

- To generate lines and curves that have a geometric width (that is, a width that is subject to transformations); see GpiModifyPath and GpiStrokePath.
- To generate lines and curves that have cosmetic width; see GpiOutlinePath. In particular, if the lines and curves are defined by characters drawn with an outline font, hollow characters are produced. Hollow characters can also be drawn outside paths, using the FATTR_SEL_OUTLINE FATTRS option with the GpiCreateLogFont function.
- To generate nonrectangular shapes to be used for clipping; see GpiSetClipPath.
- To generate shapes to be filled; see GpiFillPath.
Note: Areas can also be used for filling; see GpiBeginArea.
- To generate shapes to be converted to regions on which the region-combination function, GpiCombineRegion, can be used; see GpiPathToRegion.

GpiBeginPath —

Begin Path

There are two stages in the process of describing a path:

1. Path specification
2. Path definition.

Path Specification

A path is specified by a number of figures, within a GpiBeginPath-GpiEndPath pair. Each figure is specified by line functions, or curve functions, or both, and is separated from other figures by one of these functions:

- GpiCallSegmentMatrix
- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiFullArc
- GpiMarker
- GpiMove
- GpiPolyMarker
- GpiPop (which restores the current position)
- GpiSetCurrentPosition
- GpiSetModelTransformMatrix

A figure that is terminated by one of the functions in this list is said to be an *open* figure. A figure can also be terminated by a GpiCloseFigure function. This is said to be a *closed* figure.

A GpiBox or GpiFullArc function within a path specifies a complete closed figure. These functions must not be used within another figure specification.

GpiBeginPath initializes the path to be empty.

Path specification functions are terminated by GpiEndPath. If there are no primitives between the GpiBeginPath and GpiEndPath functions, a null path is specified. The GpiEndPath that terminates this path specification must occur within the same segment as the GpiBeginPath function.

Path specification functions can occur within a segment bracket.

Path Definition

The process of path definition causes a description of the path to be built in the currently associated device context. This description is used during any subsequent operation on the path. If the definition occurred by the drawing of a retained segment containing specification functions, these may subsequently be edited, with no effect on the path definition, until the segment is drawn again.

If the drawing mode (see GpiSetDrawingMode) is set to **draw** or **draw-and-retain**, the path is defined as it is specified. If drawing mode is **retain**, path definition does not occur until the segment containing the path specification is drawn.

When a path has been defined, the definition cannot be reopened. An attempt to redefine the path results in the definition being replaced.

As the path definition is kept in the device context, association of the presentation space with a new device context means that the definition is lost.

When it has been defined, a path can be used only in a single GpiFillPath, GpiStrokePath, GpiOutlinePath, GpiPathToRegion, or GpiSetClipPath function. Alternatively, a path can be modified once only with a GpiModifyPath function, and then used in a single GpiFillPath, GpiPathToRegion, or GpiSetClipPath function. If a path is required to be reused in a normal (not a micro) presentation space, it can be created in a retained segment (for example, using **draw-and-retain** mode [see GpiSetDrawingMode]). This segment must be drawn whenever the definition has to be recreated. This may be done even if the application is otherwise nonretained. Otherwise, the application must

GpiBeginPath – Begin Path

reissue all the individual functions to reconstruct the path whenever the definition has to be recreated.

A path definition is bound in device coordinates at the time the path is defined. If any transforms (other than the final windowing transform) are subsequently changed, they have no effect on the path itself. However, they affect the thickness if the path is to be stroked using `GpiModifyPath`, and they affect the pattern reference point if the path is to be filled with `GpiFillPath`. The transforms affect both the thickness and the pattern reference point if `GpiStrokePath` is used.

Other Remarks

Line type and line width have no effect on a path. Geometric line width takes effect if the path is stroked with `GpiModifyPath` or `GpiStrokePath`.

These functions can be used inside the path bracket (that is, between the `GpiBeginPath` function and the following `GpiEndPath` function) to define the path:

- `GpiBeginElement` (containing valid calls only)
- `GpiBox` (must specify `DRO_OUTLINE` option)
- `GpiCallSegmentMatrix`
- `GpiCharString`
- `GpiCharStringAt`
- `GpiCharStringPos`
- `GpiCharStringPosAt`
- `GpiCloseFigure`
- `GpiComment`
- `GpiElement` (containing a valid call)
- `GpiEndElement`
- `GpiFullArc` (must specify `DRO_OUTLINE` option)
- `GpiLabel`
- `GpiLine`
- `GpiMarker`
- `GpiMove`
- `GpiPartialArc`
- `GpiPointArc`
- `GpiPolyFillet`
- `GpiPolyFilletSharp`
- `GpiPolyMarker`
- `GpiPolyLine`
- `GpiPolySpline`
- `GpiPop` (if only a valid call is popped)
- `GpiSetArcParams`
- `GpiSetAttrMode`
- `GpiSetAttrs`
- `GpiSetCharAngle`
- `GpiSetCharBox`
- `GpiSetCharDirection`
- `GpiSetCharMode`
- `GpiSetCharSet`
- `GpiSetCharShear`
- `GpiSetColor`
- `GpiSetCurrentPosition`
- `GpiSetLineEnd`
- `GpiSetLineJoin`
- `GpiSetLineType`
- `GpiSetLineWidth`
- `GpiSetMarker`
- `GpiSetMarkerBox`
- `GpiSetMarkerSet`
- `GpiSetMix`
- `GpiSetModelTransformMatrix`

The `GpiCharString...` functions, `GpiQueryCharStringPos`, `GpiQueryCharStringPosAt`, and `GpiQueryTextBox` are allowed only if the current font is an outline font.

You can have no more than 1 450 straight line vertices that describe the path. Curves are decomposed into straight lines internally, and the number of resulting vertices are also subject to this limit. The same applies to outline font character strings. If solid-filled outline characters are to be drawn, it is better to do this outside a path definition. `GpiModifyPath` and `GpiStrokePath` increase the number of lines in the path, and will cause a path initially containing more than 297 straight lines to exceed the limit of 1 450.

It is not valid for this function to occur within an area definition.

GpiBeginPath —

Begin Path

Related Functions

- GpiBeginArea
- GpiCloseFigure
- GpiEndPath
- GpiFillPath
- GpiModifyPath
- GpiOutlinePath
- GpiPathToRegion
- GpiSetClipPath
- GpiStrokePath
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

Graphic Elements and Orders

Element Type: OCODE_GBPTH

Order: Begin Path

Example Code

This example uses the GpiBeginPath function to create a path. The path, an isosceles triangle, is given path identifier 1. After the path bracket is ended using GpiEndPath, a subsequent call to the GpiFillPath function draws and fills the path.

```
#define INCL_GPIPATHS          /* GPI Path functions          */
#include <os2.h>

HPS hps;                      /* presentation space handle */
POINTL ptlStart = { 0, 0 }; /* first vertex              */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices */

GpiBeginPath(hps, 1L);        /* start the path bracket */
GpiMove(hps, &ptlStart);      /* move to starting point */
GpiPolyLine(hps, 2L, ptlTriangle); /* draw two sides */
GpiCloseFigure(hps);          /* close the triangle */
GpiEndPath(hps);              /* end the path bracket */
GpiFillPath(hps, 1L, FPATH_ALTERNATE); /* draw and fill the path */
```

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

**LONG GpiBitBlt (HPS hpsTarget, HPS hpsSource, LONG ICount, PPOINTL aptlPoints,
LONG IRop, ULONG flOptions)**

This function copies a rectangle of bit-map image data.

Parameters

hpsTarget (HPS) – input

Target presentation-space handle.

hpsSource (HPS) – input

Source presentation-space handle.

ICount (LONG) – input

Point count.

Number of points specified in *aptlPoints*.

If this is 3, a source rectangle of the same size as the target rectangle is used. If it is 4, stretching or compression is performed as necessary. If compression is performed, the *flOptions* parameter determines how eliminated rows or columns are handled.

aptlPoints (PPOINTL) – input

Point array.

Array of *ICount* points, in the order **Tx1, Ty1, Tx2, Ty2, Sx1, Sy1, Sx2, Sy2**, where:

Tx1,Ty1 Specify the lower-left corner of the target rectangle in target device coordinates.

Tx2,Ty2 Specify the upper-right corner of the target rectangle in target device coordinates.

Sx1,Sy1 Specify the lower-left corner of the source rectangle in source device coordinates.

Sx2,Sy2 Specify the upper-right corner of the source rectangle in source device coordinates (not required if neither stretching nor compression is to be performed).

IRop (LONG) – input

Mixing function required.

The value of *IRop* required to achieve any given result can be determined from the following table. The final value of each bit in every pel depends on the values of the corresponding bits in the pattern (P), source (S), and the original target value (T initial). Each row of the table shows one of the 8 possible combinations of these values. For each combination, mark the desired final target value in the last column. The 8 bits in this column then show the value of the least significant byte of *IRop* required to achieve this mixing function. For example, if the required mixing function is to copy the source to the target, then the T (final) column will be the same as the S column, and so *IRop* will have the binary value 11001100, or the hexadecimal value 00CC.

P	S	T (initial)	T (final)
0	0	0	Bit 0 (least significant)
0	0	1	Bit 1
0	1	0	Bit 2
0	1	1	Bit 3
1	0	0	Bit 4
1	0	1	Bit 5
1	1	0	Bit 6
1	1	1	Bit 7 (most significant)

Mnemonic names are available for commonly used mixes:

GpiBitBlt — Bit Blt

```

ROP_SRCOPY      /* SRC                */
ROP_SRCPAINT    /* SRC OR DST            */
ROP_SRCAND      /* SRC AND DST           */
ROP_SRCINVERT   /* SRC XOR DST           */
ROP_SRCERASE    /* SRC AND NOT(DST)      */
ROP_NOTSRCCOPY  /* NOT(SRC)              */
ROP_NOTSRCERASE /* NOT(SRC) AND NOT(DST) */
ROP_MERGECOPY   /* SRC AND PAT            */
ROP_MERGEPAINT  /* NOT(SRC) OR DST       */
ROP_PATCOPY     /* PAT                   */
ROP_PATPAINT    /* NOT(SRC) OR PAT OR DST */
ROP_PATINVERT   /* DST XOR PAT           */
ROP_DSTINVERT   /* NOT(DST)              */
ROP_ZERO        /* 0                     */
ROP_ONE         /* 1                     */

```

fOptions (ULONG) — input
Options.

The options define how eliminated lines or columns are treated if a compression is performed.

Bits 15 through 31 of *fOptions* may be used for privately supported modes for particular devices.

BBO_OR The default. If compression is necessary, logical-OR the eliminated rows or columns. This is useful for white on black.

BBO_AND If compression is necessary, logical-AND the eliminated rows or columns. This is useful for black on white.

BBO_IGNORE If compression is necessary, ignore the eliminated rows or columns. This is useful for color.

Returns

Correlation and error indicators:

GPI_OK Successful completion

GPI_HITS Correlate hits

GPI_ERROR Error occurred.

Possible returns from `WinGetLastError`

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT An invalid length or count parameter was specified.

PMERR_INV_BITBLT_MIX An invalid *IRop* parameter was specified with a `GpiBitBlt` or `GpiWCBitBlt` function.

PMERR_INV_BITBLT_STYLE An invalid options parameter was specified with a `GpiBitBlt` or `GpiWCBitBlt` function.

PMERR_BITMAP_NOT_FOUND A attempt was made to perform a bit-map operation on a bit map that did not exist.

PMERR_INV_COORDINATE An invalid coordinate value was specified.

PMERR_INV_RECT An invalid rectangle parameter was specified.

PMERR_NO_BITMAP_SELECTED An attempt has been made to operate on a memory device context that has no bit map selected.

PMERR_INCORRECT_DC_TYPE An attempt was made to perform a bit-map operation on a presentation space associated with a device context of a type that is unable to support bit-map operations.

PMERR_INCOMPATIBLE_BITMAP

An attempt was made to select a bit map or perform a BitBlt operation on a device context that was incompatible with the format of the bit map.

Remarks

A rectangle of bit-map image data is copied from a bit map selected into a device context associated with the source presentation space, to a bit map selected into a device context associated with the target presentation space. Alternatively, either presentation space may be associated with a device context that specifies a suitable raster device, for example, the screen.

Note: In either case, both source and target device contexts must apply to the same physical device. It is an error if this device does not support raster operations.

Unless the device is a *banded* printer, both source and target may refer to the same presentation space. If so, the copy is nondestructive when source and target rectangles overlap.

A rectangle can be specified in device coordinates, for both source and target. These rectangles are noninclusive; that is, they include the left and lower boundaries in device space, but not the right and upper boundaries. Thus, if the lower-left maps to the same device pel as the upper-right, that rectangle is considered to be empty.

If the upper-right source point is specified, and the source and target rectangles are of different sizes, stretching, or compressing, or both, of the data occurs. *flOptions* specifies how eliminated rows or columns of bits are to be treated if compression occurs. Note that the pattern data is never stretched or compressed.

The following current attributes of the target presentation space are used (other than for converting between monochrome and color, as described below):

- Area color
- Area background color
- Pattern set
- Pattern symbol.

The color values are used in conversion between monochrome and color data. This is the only format conversion performed by this function. The conversions are:

- Output of a monochrome pattern to a color device.

In this instance, the pattern is converted first to a color pattern using the current area colors:

- source 1s → area foreground color
- source 0s → area background color.

- Copying from a monochrome bit map to a color bit map (or device).

The source bits are converted as follows:

- source 1s → image foreground color
- source 0s → image background color.

- Copying from a color bit map to a monochrome bit map (or device).

- source pels that are the source image background color → image background color.
- all other pels → image foreground color.

Note: In all of the above instances (except where the source image background color is used) it is the attributes of the *target* presentation space that are used.

If the mix (*IRop*) does not call for a pattern, the pattern set and pattern symbol are not used. If it does not require a source (this is not valid when *flOptions* is in the range 1 through 3), *hpsSource* is not required and must be null. **Sx1,Sy1** is also ignored in this instance.

GpiBitBlt — Bit Blt

Neither the source nor the pattern is required when a bit map, or part of a bit map, is to be cleared to a particular color.

If the mix does require both source and pattern, a three-way operation is performed.

If a pattern is required, dithering may be performed for solid patterns in a color that is not available on the device; see GpiSetPattern.

If any of the source data is not available (when, for example, the source presentation space is connected to a screen window, and the source rectangle is not totally visible), the contents of the unavailable parts are undefined. This can be checked with GpiRectVisible before calling this function.

This function is independent of drawing mode (see GpiSetDrawingMode); the effect always occurs immediately, and it is not retained even if the drawing mode is **draw-and-retain** or **retain**. Its effect, however, is recorded in a metafile, but note that this is successful only if the metafile is replayed on a similar device, with **draw** drawing mode.

The current position in both source and target presentation spaces is unchanged by this function.

Note: This function must not be used when creating SAA-conforming metafiles; see “Metafile Restrictions” on page G-1.

Related Functions

- DevQueryCaps
- DevOpenDC
- GpiCreateBitmap
- GpiDeleteBitmap
- GpiDrawBits
- GpiLoadBitmap
- GpiQueryBitmapBits
- GpiQueryBitmapDimension
- GpiQueryBitmapHandle
- GpiQueryBitmapParameters
- GpiQueryDeviceBitmapFormats
- GpiSetBitmap
- GpiSetBitmapBits
- GpiSetBitmapDimension
- GpiSetBitmapId
- GpiWCBlt
- WinDrawBitmap
- WinGetSysBitmap

Example Code

This example uses GpiBitBlt to copy a bit map from one presentation space to another. Two presentation spaces are created: one associated with a memory context, and the other associated with a screen context. The function copies the memory context bit map that is 100 pels wide and 100 pels high into a 50-by-50-pel rectangle at the location (300,400) on the screen, thereby causing the bit map to be visible in the window. Since the raster operation is ROP_SRCCOPY, GpiBitBlt replaces the image previously in the target rectangle. The function compresses the bit map to fit the new rectangle by discarding extra rows and columns as specified by the BBO_IGNORE option.

```
#define INCL_GPIBITMAPS      /* Bit map functions          */
#define INCL_DEV             /* Device Function definitions */
#define INCL_GPICONTROL      /* GPI control Functions      */
#define INCL_WINWINDOWMGR    /* Window Manager Functions    */
#include <os2.h>

HAB    hab;          /* anchor-block handle          */
HPS    hpsMemory;     /* presentation-space handle     */
HPS    hpsScreen;     /* presentation-space handle     */
HDC    hdcScreen;     /* Device-context handle         */
HDC    hdcMemory;     /* Device-context handle         */
SIZEL  sizl={0, 0};   /* use same page size as device */
/* context data structure */
DEVOPENSTRUC dop = {0L, "DISPLAY", NULL, 0L, 0L, 0L, 0L, 0L};
POINTL aptl[4] = {
    300, 400,          /* lower-left corner of target   */
    350, 450,          /* upper-right corner of target  */
    0, 0,              /* lower-left corner of source   */
    100, 100 };        /* upper-right corner of source  */
HWND   hwnd;

/* create memory device context and presentation space, associating
   DC with the PS */
hdcMemory = DevOpenDC(hab, OD_MEMORY, "", 5L, (PDEVOPENDATA)&dop,
    NULLHANDLE);
hpsMemory = GpiCreatePS(hab, hdcMemory, &sizl, GPIA_ASSOC
    | PU_PELS);

/* create window device context and presentation space, associating
   DC with the PS */
hdcScreen = WinOpenWindowDC(hwnd); /* Open window device context */
hpsScreen = GpiCreatePS(hab, hdcScreen, &sizl, PU_PELS | GPIF_LONG
    | GPIA_ASSOC);

/*
 * . get bit map, associate bit map with memory device context,
 * .   draw into bit map
 * .
 */

/* display the bit map on the screen by copying it from the memory
   device context into the screen device context */
GpiBitBlt(hpsScreen, hpsMemory, 4L, aptl, ROP_SRCCOPY, BBO_IGNORE);
```


GpiBox — Box

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

LONG GpiBox (HPS hps, LONG IControl, PPOINTL pptlPoint, LONG IHRound, LONG IVRound)

This function draws a rectangular box with the current position and a specified position at diagonally opposite corners.

Parameters

hps (HPS) — input
Presentation-space handle.

IControl (LONG) — input
Outline and fill control.

Specifies if the interior of the box is to be filled, and if the outline is to be drawn:

DRO_FILL Fill interior

DRO_OUTLINE Draw outline

DRO_OUTLINEFILL Draw outline and fill interior.

pptlPoint (PPOINTL) — input
Corner point.

The coordinates of the corner that is diagonally opposite to the current position.

IHRound (LONG) — input
Corner-rounding control.

Horizontal length of the *full* axis of the ellipse that is used for rounding at each corner.

IVRound (LONG) — input
Corner-rounding control.

Vertical length of the *full* axis of the ellipse that is used for rounding at each corner.

Returns

Correlation and error indicators:

GPI_OK Successful

GPI_HITS Correlate hits

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_BOX_CONTROL An invalid control parameter was specified with GpiBox.

PMERR_INV_COORDINATE An invalid coordinate value was specified.

PMERR_INV_BOX_ROUNDING_PARM An invalid corner rounding control parameter was specified with GpiBox.

Remarks

The sides of the box are parallel to the world coordinate x- and y-axes.

The four corners of the box can be rounded with a quarter ellipse. The size of this ellipse is specified by *IHRound* and *IVRound*. If *IHRound* equals *IVRound*, the corners of the box are rounded with a quarter circle.

If either *IHRound* or *IVRound* is zero, no rounding occurs.

If the current position is (x0,y0) and *pptlPoint* is set to (x1,y1), the box is drawn from (x0,y0) to (x1,y0) to (x1,y1) to (x0,y1) to (x0,y0). The direction of drawing is significant in area winding mode; see *GpiBeginArea*.

The current position is unchanged by this function.

Either the outline of the box, or its interior, or both, can be drawn.

If this function occurs within an area or path definition, it generates a complete closed figure (*DRO_OUTLINE* must be specified). It must not occur within any other figure definition.

If correlation is in force, a hit always results if the pick aperture intersects the box boundary. However, if the pick aperture lies wholly within the box, a hit only occurs if the interior is being drawn (*DRO_FILL* or *DRO_OUTLINEFILL*).

Related Functions

- *GpiBox*
- *GpiQueryCurrentPosition*
- *GpiSetCurrentPosition*
- *GpiSetLineJoin*
- *GpiSetLineType*
- *GpiSetLineWidth*
- *GpiSetLineWidthGeom*
- *GpiPop*
- *GpiSetAttrMode*
- *GpiSetAttrs*
- *GpiSetDefAttrs*
- *GpiSetBackColor*
- *GpiSetBackMix*
- *GpiSetColor*
- *GpiSetMix*

Graphic Elements and Orders

Element Type: **OCODE_GCBOX**

Order: **Box at Current Position**

GpiBox — Box

Example Code

This example calls GpiBox to draw a series of rounded boxes, one inside another.

```
#define INCL_GPIPRIMITIVES    /* GPI primitive functions */
#include <os2.h>

HPS hps;                      /* presentation space handle */
POINTL ptl = { 100, 100 };
SHORT i;

for (i = 0; i < 5; i++)
    GpiBox(hps,                /* handle to a presentation space */
           DRO_OUTLINE,        /* draw the box outline */
           &ptl,               /* address of the corner */
           i * 10L,            /* horizontal corner radius */
           i * 10L);           /* vertical corner radius */
```

GpiCallSegmentMatrix – Call Segment Matrix

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
```

**LONG GpiCallSegmentMatrix (HPS hps, LONG ISegment, LONG ICount,
PMATRIXLF pmatIfArray, LONG IOptions)**

This function calls a segment and applies an instance transform to it.

Parameters

hps (HPS) – input
Presentation-space handle.

ISegment (LONG) – input
Identifier of segment to be called.

This must be greater than 0.

The segment must not be a chained segment.

ICount (LONG) – input
Number of elements.

The number of elements of *pmatIfArray* to be examined, starting from the beginning of the structure. If *ICount* is less than 9, the remaining elements default to the corresponding elements of the identity matrix. If *ICount* = 0, the identity matrix is used.

pmatIfArray (PMATRIXLF) – input
Instance transform matrix.

The third, sixth, and ninth elements, when specified, must be 0, 0, and 1, respectively.

IOptions (LONG) – input
Transformation options.

Specify how the transform defined by the *pmatIfArray* parameter should be used to modify the existing current model transform for the duration of the function. The existing transform is the concatenation, in the current function context, of the instance, segment, and model transforms, from the root segment downwards.

TRANSFORM_REPLACE The previous model transform is discarded and replaced by the specified transform.

TRANSFORM_ADD The specified transform is combined with the existing model transform. The existing transform precedes the new transform. This option is most useful for incremental updates to transforms.

TRANSFORM_PREEMPT The specified transform is combined with the existing model transform. The new transform precedes the existing transform.

Returns

Correlation and error indicators:

GPI_OK Successful

GPI_HITS Correlate hits

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

GpiCallSegmentMatrix —

Call Segment Matrix

PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_SEG_NAME	An invalid segment identifier was specified.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_INV_MATRIX_ELEMENT	An invalid transformation matrix element was specified.
PMERR_INV_TRANSFORM_TYPE	An invalid options parameter was specified with a transform matrix function.
PMERR_CALLED_SEG_NOT_FOUND	An attempt was made to call a segment that did not exist.
PMERR_CALLED_SEG_IS_CHAINED	An attempt was made to call a segment that has a chained attribute set.
PMERR_CALLED_SEG_IS_CURRENT	An attempt was made to call a segment that is currently open.
PMERR_SEG_CALL_STACK_EMPTY	A call stack empty condition was detected when attempting a pop function during GpiPop or segment drawing.

Remarks

The instance transform specified is a model transform that is used to modify the current model transform, in a way that depends upon the value of the *IOptions* parameter, before calling the segment. This new transform applies only to the called segment. On return, it is reset to the model transform in operation before the function was called.

The transform is specified as a one-dimensional array of elements, being the first *ICount* elements of a 3-row by 3-column matrix ordered by rows. The order of the elements is:

Matrix

Array

a	b	0
c	d	0
e	f	1

(a,b,0,c,d,0,e,f,1)

A point with coordinates (x,y) is transformed to the point

$(a*x + c*y + e, b*x + d*y + f)$

The called segment must have a unity transform for the viewing transform (see *GpiSetViewingTransformMatrix*).

If scaling values greater than unity are given (which only applies if the presentation space coordinate format as set by the *GpiCreatePS* function is *GPIF_LONG*), it is possible for the combined effect of this and any other relevant transforms to exceed fixed-point implementation limits. This causes an error.

Related Functions

- *GpiCloseSegment*
- *GpiCorrelateSegment*
- *GpiDeleteSegment*
- *GpiDeleteSegments*
- *GpiDrawSegment*
- *GpiErrorSegmentData*

GpiCallSegmentMatrix – Call Segment Matrix

- GpiOpenSegment
- GpiQueryInitialSegmentAttrs
- GpiQuerySegmentAttrs
- GpiQuerySegmentNames
- GpiQuerySegmentPriority
- GpiSetInitialSegmentAttrs
- GpiSetSegmentAttrs
- GpiSetSegmentPriority
- GpiSetSegmentTransformMatrix

Graphic Elements and Orders

Element Type: OCODE_GCALLS

Order: Push and Set Model Transform

Order: Call Segment

Order: Pop

Example Code

This example calls the GpiCallSegmentMatrix function to draw a segment three times. Each time the segment is drawn, the instance transformation doubles in size. The result is three triangles with the last triangle twice the size of the second, and the second twice the size of the first.

```
#define INCL_GPITRANSFORMS    /* GPI Transform functions    */
#define INCL_GPISEGMENTS     /* Segment functions    */
#define INCL_GPIPRIMITIVES   /* GPI primitive functions */
#include <os2.h>

HPS    hps;
USHORT i;
POINTL ptlStart = { 0, 0 }; /* first vertex */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices */
MATRIXLF matlfInstance = { MAKEFIXED(1, 0), MAKEFIXED(0, 0), 0,
                           MAKEFIXED(0, 0), MAKEFIXED(1, 0), 0,
                           0, 0, 1 };

GpiOpenSegment(hps, 1L); /* opens segment */
GpiMove(hps, &ptlStart); /* moves to start point (0, 0) */
GpiPolyLine(hps, 3L, ptlTriangle); /* draws triangle */
GpiCloseSegment(hps); /* closes segment */

for (i = 0; i < 3; i++)
{
    /*
     * Draw the segment after adding the matrix to the model
     * transformation.
     */

    GpiCallSegmentMatrix(hps, 1L, 9, &matlfInstance, TRANSFORM_ADD);
    matlfInstance.fxM11 *= 2;
    matlfInstance.fxM22 *= 2;
}
```

GpiCharString – Character String

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

LONG GpiCharString (HPS hps, LONG ICount, PCH pchString)

This function draws a character string starting at the current position.

Parameters

hps (HPS) – input
Presentation-space handle.

ICount (LONG) – input
Number of bytes in the string.
The maximum number is 512.

pchString (PCH) – input
Characters to be drawn.

Returns

Correlation and error indicators:

GPI_OK Successful

GPI_HITS Correlate hits

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_FONT_AND_MODE_MISMATCH	An attempt was made to draw characters with a character mode and character set that are incompatible. For example, the character specifies an image/raster font when the mode calls for a vector/outline font.

Remarks

Each character in the string is positioned so that its character reference point is at the current position. The current position is advanced after each character is drawn to give the position for the next character.

The characters in the character string are selected from the current character set. The font from which the characters are selected depends on the current character mode. For a description of which fonts are used for each of the possible modes, see GpiSetCharMode.

The degree to which approximation of the position and size of characters is allowed, and also the area used during correlation of the character string, is controlled by the character-mode attribute.

After the string has been drawn, the current position is set to the end of the character string. This is the point at which the next character would have been drawn, had it existed.

Related Functions

- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt
- GpiQueryDefCharBox
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

Graphic Elements and Orders

Element Type: OCODE_GCCHSTM

Order: Character String Move at Current Position

Example Code

This example uses the GpiCharString function to draw the string 'Hello'. The GpiMove function moves the current position to (100,100) so that the string starts there.

```
#define INCL_GPIPRIMITIVES    /* GPI primitive functions */
#include <os2.h>

HPS hps;                    /* presentation space handle */
POINTL ptlStart;           /* beginning of string */

ptlStart.x = 100L;
ptlStart.y = 100L;

/* Start string at (100, 100). */

GpiMove(hps, &ptlStart);

/* Draw the 5-character string. */

GpiCharString(hps, 5L, "Hello");
```


GpiCharStringAt – Character String At

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

LONG GpiCharStringAt (HPS hps, PPOINTL pptlPoint, LONG ICount, PCH pchString)

This function draws a character string starting at a specified position.

Parameters

hps (HPS) – input

Presentation-space handle.

pptlPoint (PPOINTL) – input

Starting position.

Defines, in world coordinates, the position at which the first character in the string is to be placed.

ICount (LONG) – input

Number of bytes in the string.

The maximum number is 512.

pchString (PCH) – input

Characters to be drawn.

Returns

Correlation and error indicators:

GPI_OK Successful

GPI_HITS Correlate hits

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE

An invalid coordinate value was specified.

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

PMERR_FONT_AND_MODE_MISMATCH

An attempt was made to draw characters with a character mode and character set that are incompatible. For example, the character specifies an image/raster font when the mode calls for a vector/outline font.

Remarks

The function GpiCharStringAt (hps, point, count, string) is equivalent to:

GpiMove (hps, point)

GpiCharString (hps, count, string)

Each character in the string is positioned so that its character reference point is at the current position. The current position is advanced after each character is drawn to give the position for the next character.

GpiCharStringAt – Character String At

The font from which the characters in the character string are selected depends on the current character mode. For a description of which fonts are used for each of the possible modes, see GpiSetCharMode.

The degree to which approximation of the position and size is allowed, and also the area used during correlation of the character string, is controlled by the character-mode attribute.

After the string has been drawn, the current position is set to the end of the character string. This is the point at which the next character would have been drawn, had it existed.

Related Functions

- GpiCharString
- GpiCharStringPos
- GpiCharStringPosAt
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt
- GpiQueryDefCharBox
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

Graphic Elements and Orders

Element Type: OCODE_GCHSTM

Order: Character String Move at Given Position

GpiCharStringAt – Character String At

Example Code

This example uses the GpiCharStringAt function to draw the string "Hello" starting at the position (100,100). It then uses the GpiMove and GpiCharString functions to draw the same string at exactly the same position.

```
#define INCL_GPIPRIMITIVES    /* GPI primitive functions    */
#include <os2.h>

HPS hps;                    /* presentation space handle    */
POINTL ptlStart;

ptlStart.x = 100L;
ptlStart.y = 100L;

/* Draw the string "Hello" at (100, 100). */
GpiCharStringAt(hps, &ptlStart, 5, "Hello");

/* These two calls are identical to the one above. */
GpiMove(hps, &ptlStart);
GpiCharString(hps, 5L, "Hello");
```

GpiCharStringPos – Character String Position

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiCharStringPos (HPS hps, PRECTL prclRect, ULONG flOptions, LONG ICount, PCH pchString, PLONG alAdx)

This function draws a character string starting at the current position, with formatting options.

Parameters

hps (HPS) – input

Presentation-space handle.

prclRect (PRECTL) – input

Rectangle structure.

Defines, in world coordinates, the two corners of the rectangle that defines the background of the characters. It is ignored unless CHS_OPAQUE or CHS_CLIP is specified.

flOptions (ULONG) – input

Formatting options.

Option flags that can be used in combination:

CHS_OPAQUE

Background of characters is defined by the rectangle specified by *prclRect*. The rectangle is to be shaded (with background color and overpaint) before drawing.

CHS_VECTOR

Increments vector (*alAdx*) is supplied. If zero, *alAdx* is ignored.

CHS_LEAVEPOS

Leave the current position at the start of the string. If not set, the current position is moved to the position at which the next character would have been drawn, had there been one.

CHS_CLIP

Clip the string to the rectangle.

CHS_UNDERSCORE

Underscore the characters. See FATTR_SEL_UNDERSCORE on page A-37 in the FATTRS on page A-36 datatype.

CHS_STRIKEOUT

Overstrike the characters. See FATTR_SEL_STRIKEOUT in the FATTRS datatype.

Other bits are reserved and must be zero.

ICount (LONG) – input

Number of bytes in the string.

The maximum number is 512.

pchString (PCH) – input

Characters to be drawn.

alAdx (PLONG) – input

Increment values.

Vector of increment values, in world coordinates. Any negative values are treated as if they were zero.

GpiCharStringPos — Character String Position

Returns

Correlation and error indicators:

GPI_OK	Successful
GPI_HITS	Correlate hits
GPI_ERROR	Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_CHAR_POS_OPTIONS	An invalid options parameter was specified with GpiCharStringPos or GpiCharStringPosAt.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_INV_RECT	An invalid rectangle parameter was specified.
PMERR_FONT_AND_MODE_MISMATCH	An attempt was made to draw characters with a character mode and character set that are incompatible. For example, the character specifies an image/raster font when the mode calls for a vector/outline font.

Remarks

A vector of increments can be specified, allowing control over the positioning of each character after the first. This vector consists of distances measured in world coordinates (along the baseline for left-to-right and right-to-left character directions, and along the shearline for top-to-bottom and bottom-to-top character directions). Increment i is the distance of the reference point of character $i+1$ from the reference point of character i . The last increment may be needed to update the current position.

These increments, when specified, set the widths of each character.

A further option allows a rectangle to be specified that can be used as the background of the string instead of the normal background. This rectangle is painted using the current character background color and an overpaint mix (unless this is in a dynamic segment, when leave-alone is used). Both corners of the rectangle are specified, so that the rectangle is positioned independently of the current position. Points on the borders of the rectangle are considered to be included within the rectangle.

Clipping of the string to the rectangle is also allowed. This is independent of whether the rectangle is actually drawn.

The current position can be updated to the point at which the next character would have been drawn, had there been one, or it can be left at the start of the string.

Related Functions

- GpiCharString
- GpiCharStringAt
- GpiCharStringPosAt
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt
- GpiQueryDefCharBox
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode

GpiCharStringPos – Character String Position

- GpiSetCharSet
- GpiSetCharShear
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

Graphic Elements and Orders

Element Type: ETYPE_GCCHSTE

Order: Character String Extended at Current Position

Example Code

This example uses GpiCharStringPos to display '13 Characters', starting at position 10,10 and clipped to a 100x100 rectangle in the lower left corner.

```
#define INCL_GPIPRIMITIVES      /* GPI Primitive functions */
#include <os2.h>

LONG lHits;          /* correlation/error indicator */
HPS hps;             /* Presentation-space handle */
POINTL pptlStart = {10L,10L};
                    /* Starting position */
RECTL prclRect = {0L,0L,100L,100L};
                    /* Rectangle structure */
ULONG flOptions;     /* Formatting options */
LONG lCount;         /* Number of bytes in the string */
char pchString[25];  /* Characters to be drawn */

GpiMove(hps, &pptlStart);

flOptions = CHS_CLIP; /* clip text to rectangle */
lCount = 13;
strcpy(pchString,"13 characters");

/* draw the string */
lHits = GpiCharStringPos(hps, &prclRect, flOptions, lCount,
                        pchString, NULL);
```

GpiCharStringPosAt – Character String Position At

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiCharStringPosAt (HPS hps, PPOINTL pptlStart, PRECTL prclRect, ULONG flOptions, LONG lCount, PCH pchString, PLONG alAdx)

This function draws a character string starting at a specified position, with formatting options.

Parameters

hps (HPS) – input

Presentation-space handle.

pptlStart (PPOINTL) – input

Starting position.

prclRect (PRECTL) – input

Rectangle structure.

Defines, in world coordinates, the two corners of the rectangle that defines the background of the characters. It is ignored unless CHS_OPAQUE or CHS_CLIP is selected.

flOptions (ULONG) – input

Formatting options.

Option flags that can be used in combination:

CHS_OPAQUE

Background of characters is defined by the rectangle specified by *prclRect*. The rectangle is to be shaded (with background color and overpaint) before drawing.

CHS_VECTOR

Increments vector (*alAdx*) is supplied. If 0, *alAdx* is ignored.

CHS_LEAVEPOS

If set, current position is unchanged by this function. If not set, current position is moved to the position at which the next character would have been drawn, had there been one.

CHS_CLIP

Clip the string to the rectangle.

CHS_UNDERSCORE

Underscore the characters. See FATTR_SEL_UNDERSCORE in the FATTRS datatype.

CHS_STRIKEOUT

Overstrike the characters. See FATTR_SEL_STRIKEOUT in the FATTRS datatype.

Other bits are reserved and must be zero.

lCount (LONG) – input

Number of bytes in the string.

The maximum number is 512.

pchString (PCH) – input

Character string.

alAdx (PLONG) – input

Increment values.

Vector of increment values, in world coordinates. Any negative values are treated as if they were zero.

GpiCharStringPosAt – Character String Position At

Returns

Correlation and error indicators:

GPI_OK	Successful
GPI_HITS	Correlate hits
GPI_ERROR	Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_CHAR_POS_OPTIONS	An invalid options parameter was specified with GpiCharStringPos or GpiCharStringPosAt.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.
PMERR_INV_RECT	An invalid rectangle parameter was specified.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_FONT_AND_MODE_MISMATCH	An attempt was made to draw characters with a character mode and character set that are incompatible. For example, the character specifies an image/raster font when the mode calls for a vector/outline font.

Remarks

A vector of increments can be specified, allowing control over the position of each character after the first. This vector consists of distances measured in world coordinates (along the baseline for left-to-right and right-to-left character directions, and along the shearline for top-to-bottom and bottom-to-top character directions). Increment i is the distance of the reference point (for example, lower left corner) of character $i+1$ from the reference point of character i . The last increment may be needed to update the current position.

These increments, if specified, set the widths of each character.

A further option allows a rectangle to be specified that can be used as the background of the string instead of the normal background. This rectangle is painted using the current character background color and an overpaint mix (unless this is in a dynamic segment, when leave-alone is used). Both corners of the rectangle are specified, so that the rectangle is positioned independently of current position. Points on the borders of the rectangle are considered to be included within the rectangle.

Clipping of the string to the rectangle is also allowed. This is independent of whether the rectangle is actually drawn.

Current position can be updated to the point at which the next character would have been drawn, had there been one, or it can be left at the start of the string.

GpiCharStringPosAt — Character String Position At

Related Functions

- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt
- GpiQueryDefCharBox
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

Graphic Elements and Orders

Element Type: ETYPE_GCHSTE

Order: Character String Extended at Given Position

Example Code

This example uses GpiCharStringPosAt to display '13 Characters', starting at position 10,10 and clipped to a 100x100 rectangle in the lower left corner.

```
#define INCL_GPIPRIMITIVES      /* GPI Primitive functions      */
#include <os2.h>

LONG  lHits;          /* correlation/error indicator      */
HPS   hps;            /* Presentation-space handle      */
POINTL pptlStart = {10L,10L}; /* Starting position */
RECTL rclRect = {0L,0L,100L,100L}; /* Rectangle structure */
ULONG flOptions;      /* Formatting options */
LONG  lCount;         /* Number of bytes in the string */
char  pchString[14];  /* Characters to be drawn */

flOptions = CHS_CLIP; /* clip text to rectangle */
lCount = 13;
strcpy(pchString,"13 characters");

lHits = GpiCharStringPosAt(hps, &pptlStart, &rclRect, flOptions,
                          lCount, pchString, NULL);
```

GpiCloseFigure – Close Figure

```
#define INCL_GPIPATHS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiCloseFigure (HPS hps)

This function closes a figure within a path specification.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Success indicator:
TRUE Successful completion
FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.

Remarks

The current figure is closed by a line drawn to the start point of the figure.

This function need not be used if the path is to be filled (see GpiFillPath), or used as a clip path (see GpiSetClipPath), as any figures in the path that have not been closed are automatically closed at that time. It should be used, however, for any closed figures within paths that are subsequently to be stroked by GpiModifyPath or GpiStrokePath.

This function must not be used outside a path specification. In particular, it must not be used within an area.

Related Functions

Prerequisite Functions

- GpiBeginPath

Other Related Functions

- GpiEndPath
- GpiModifyPath
- GpiStrokePath

Graphic Elements and Orders

Element Type: OCODE_GCFIG

Order: Close Figure

GpiCloseFigure — Close Figure

Example Code

This example uses the GpiCloseFigure function to close a triangle drawn in a path bracket. The triangle starts at (0,0), and as the current position just before the GpiCloseFigure is (200,0), the function closes the triangle by drawing a line from (200,0) to (0,0).

```
#define INCL_GPIPATHS          /* GPI Path functions */
#include <os2.h>

HPS hps;                      /* presentation space handle */
POINTL ptlStart = { 0, 0 };
POINTL ptlPoints[] = { 100, 100, 200, 0 };

GpiBeginPath(hps, 1L);        /* start the path bracket */
GpiMove(hps, &ptlStart);      /* move to starting point */
GpiPolyLine(hps, 2L, ptlPoints); /* draw two sides */
GpiCloseFigure(hps);          /* close the triangle */
GpiEndPath(hps);              /* end the path bracket */
```

GpiCloseSegment – Close Segment

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiCloseSegment (HPS hps)

This function closes the current segment.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from GetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION

An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_NOT_IN_SEG

An attempt was made to end a segment using GpiCloseSegment while not in a segment bracket.

PMERR_PATH_INCOMPLETE

An attempt was made to open or close a segment either directly or during segment drawing, or to issue GpiAssociate while there is an open path bracket.

PMERR_AREA_INCOMPLETE

Either:

- A segment has been opened, closed, or drawn.
- GpiAssociate was issued while an area bracket was open.
- A drawn segment has opened an area bracket and ended without closing it.

Remarks

Closing a segment does not delete the segment or affect the graphics primitives that are drawn.

Any attributes that have been preserved (see the AM_PRESERVE option of GpiSetAttrMode) are popped (restored) when the GpiCloseSegment function is issued in **draw** or **draw-and-retain** modes, and at the end of the segment when the segment is subsequently drawn in **draw-and-retain** or **retain** modes (see GpiSetDrawingMode).

If an area or path is open when a segment is closed, the area or path is terminated. When the drawing mode is **draw** or **draw-and-retain**, a warning is given, but the close processing continues. No warning is given for **retain** mode. If a retained segment with an open area or path is drawn, an error occurs.

If an element bracket is open when a segment is closed, the element bracket is first closed automatically.

GpiCloseSegment —

Close Segment

If this function is followed by primitives or attributes, without first opening a segment, the following may or may not have been reset to their default values:

- Current attribute values and arc parameters
- Current tag
- Current model transform
- Current position
- Current clip path and viewing limits.

Any such quantity can be assumed to contain its default value only if it is known either that it has not been changed from the default, or that last time it was changed, it was set to its default value. An application should not be written to depend on the values of these quantities immediately after GpiCloseSegment.

Subsequent primitives, not preceded by an GpiOpenSegment function, are not retained, irrespective of the current drawing mode.

The current viewing transform, however, is guaranteed to be reset to unity for primitives outside segments.

Related Functions

Prerequisite Functions

- GpiOpenSegment

Other Related Functions

- GpiCallSegmentMatrix
- GpiCorrelateSegment
- GpiDeleteSegment
- GpiDeleteSegments
- GpiDrawSegment
- GpiErrorSegmentData
- GpiQueryInitialSegmentAttrs
- GpiQuerySegmentAttrs
- GpiQuerySegmentNames
- GpiQuerySegmentPriority
- GpiSetInitialSegmentAttrs
- GpiSetSegmentAttrs
- GpiSetSegmentPriority

Example Code

This example uses the GpiCloseSegment function to close a segment. The GpiOpenSegment opens the segment; GpiMove and GpiPolyLine draw a triangle.

```
#define INCL_GPISSEGMENTS      /* Segment functions          */
#include <os2.h>

HPS hps;                      /* presentation space handle */
POINTL ptlStart = { 0, 0 }; /* first vertex               */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices */

GpiOpenSegment(hps, 1L);      /* open the segment          */
GpiMove(hps, &ptlStart);      /* move to start point (0,0) */
GpiPolyLine(hps, 3L, ptlTriangle); /* draw triangle            */
GpiCloseSegment(hps);         /* close the segment         */
```

GpiCombineRegion – Combine Region

```
#define INCL_GPIREGIONS /* Or use INCL_GPI or INCL_PM */
```

**LONG GpiCombineRegion (HPS hps, HRGN hrgnDest, HRGN hrgnSrc1, HRGN hrgnSrc2,
LONG IMode)**

This function combines two regions.

Parameters

hps (HPS) – input

Presentation-space handle.

The regions must be owned by the device identified by the currently associated device context.

hrgnDest (HRGN) – input

Handle of destination.

hrgnSrc1 (HRGN) – input

Handle of first source region.

hrgnSrc2 (HRGN) – input

Handle of second source region.

IMode (LONG) – input

Method of combination:

CRGN_OR Union of *hrgnSrc1* and *hrgnSrc2*

CRGN_COPY *hrgnSrc1* only (*hrgnSrc2* ignored)

CRGN_XOR Symmetric difference of *hrgnSrc1* and *hrgnSrc2*

CRGN_AND Intersection of *hrgnSrc1* and *hrgnSrc2*

CRGN_DIFF *hrgnSrc1* and not (*hrgnSrc2*).

Returns

Complexity of resulting region and error indicators:

RGN_NULL Null region

RGN_RECT Rectangular region

RGN_COMPLEX Complex region

RGN_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HRGN

An invalid region handle was specified.

PMERR_REGION_IS_CLIP_REGION

An attempt was made to perform a region operation on a region that is selected as a clip region.

PMERR_INV_REGION_MIX_MODE

An invalid mode parameter was specified with GpiCombineRegion.

PMERR_HRGN_BUSY

An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

GpiCombineRegion – Combine Region

Remarks

Source and destination regions must all be of the same device class. The destination region can be one of the source regions.

An error is raised if any of the specified regions are currently selected as the clip region (by GpiSetClipRegion).

Related Functions

- GpiCreateRegion
- GpiDestroyRegion
- GpiEqualRegion
- GpiOffsetRegion
- GpiPaintRegion
- GpiPtInRegion
- GpiQueryRegionBox
- GpiQueryRegionRects
- GpiRectInRegion
- GpiSetRegion

Example Code

This example uses the GpiCombineRegion function to create a complex region consisting of everything in two rectangles except where they overlap.

```
#define INCL_GPIREGIONS          /* Region functions          */
#include <os2.h>

HPS hps;                        /* presentation space handle */
HRGN hrgn1, hrgn2, hrgn3;
RECTL rc1Rect1 = { 0, 0, 100, 100 };
RECTL rc1Rect2 = { 50, 50, 200, 200 };

/* create first region */
hrgn1 = GpiCreateRegion(hps, 1L, &rc1Rect1);
/* create second region */
hrgn2 = GpiCreateRegion(hps, 1L, &rc1Rect2);
/* create empty region */
hrgn3 = GpiCreateRegion(hps, 0L, NULL);

/* Combine first and second regions, replacing the empty region. */
GpiCombineRegion(hps, hrgn3, hrgn1, hrgn2, CRGN_XOR);
```

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiComment (HPS hps, LONG lLength, PBYTE pbData)

This function adds a comment to the current segment.

Parameters

hps (HPS) – input

Presentation-space handle.

lLength (LONG) – input

Data length.

The length of *pbData* in bytes. *lLength* must not be greater than 255.

pbData (PBYTE) – input

Comment string.

No conversion of any kind is performed on the data.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

Remarks

An application can use this function to store some data of its own in the segment if the drawing mode (see GpiSetDrawingMode) is set to **retain** or **draw-and-retain**. It has no effect on drawing. The data can subsequently be retrieved by the application using GpiQueryElement or GpiGetData.

Graphic Elements and Orders

Element Type: **OCODE_GCOMT**

Order: **Comment**

GpiComment — Comment

Example Code

This example uses the GpiComment function to comment the contents of a segment.

```
#define INCL_GPIPRIMITIVES    /* GPI primitive functions    */
#define INCL_GPISEGMENTS     /* Segment functions    */
#include <os2.h>

HPS hps;                    /* presentation space handle    */
POINTL ptlStart = { 0, 0 }; /* first vertex                */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices    */

GpiOpenSegment(hps, 0L);    /* open the segment    */
GpiComment(hps, 18L, "Start point (0, 0)");
GpiMove(hps, &ptlStart);
GpiComment(hps, 13L, "Draw triangle");
GpiPolyLine(hps, 3L, ptlTriangle);
GpiCloseSegment(hps);      /* close the segment    */
```

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
```

```
BOOL GpiConvert (HPS hps, LONG ISrc, LONG ITarg, LONG ICount, PPOINTL aptlPoints)
```

This function converts an array of coordinate pairs from one coordinate space to another.

Parameters

hps (HPS) – input
Presentation-space handle.

ISrc (LONG) – input
Source coordinate space.

ITarg (LONG) – input
Target coordinate space.

ICount (LONG) – input
Number of coordinate pairs in *aptlPoints*.

aptlPoints (PPOINTL) – input/output
Array of coordinate pair structures.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_INV_COORD_SPACE	An invalid source or target coordinate space parameter was specified with GpiConvert.
PMERR_COORDINATE_OVERFLOW	An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

Remarks

This function replaces each coordinate pair in *aptlPoints* with the converted values.

Valid values for the *ISrc* and *ITarg* parameters are:

CVTC_WORLD World coordinates

CVTC_MODEL Model space

CVTC_DEFAULTPAGE Page space before default viewing transform

CVTC_PAGE Page space after default viewing transform

CVTC_DEVICE Device space.

Conversions involving either world coordinates or model space should not be performed if the drawing mode (see GpiSetDrawingMode) is **retain**.

GpiConvert — Convert

Related Functions

- GpiCreatePS
- GpiSetDefaultViewMatrix
- GpiSetModelTransformMatrix
- GpiSetPageViewport
- GpiSetSegmentTransformMatrix
- GpiSetViewingTransformMatrix

Example Code

This example uses the GpiConvert function to convert the coordinates of the mouse pointer to the corresponding coordinates in world space. The system passes mouse coordinates to a window procedure in the WM_MOUSEMOVE message. The coordinates are device coordinates. After the coordinates are converted, the GpiMove uses them to move to a new location in world space.

```
#define INCL_GPITRANSFORMS    /* GPI Transform functions    */
#define INCL_GPIPRIMITIVES   /* GPI primitive functions */
#include <os2.h>

MPARAM mp1;
HPS     hps;
POINTL  pt1;

case WM_MOUSEMOVE:
    pt1.x = (LONG) SHORT1FROMMP(mp1);
    pt1.y = (LONG) SHORT2FROMMP(mp1);
    GpiConvert(hps, CVTC_DEVICE, CVTC_WORLD, 1L, &pt1);
    GpiMove(hps, &pt1);
```

GpiConvertWithMatrix – Convert with Matrix

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiConvertWithMatrix (HPS hps, LONG ICount, PPOINTL aptlPoints, LONG ICount, PMATRIXLF pmatlfArray)

This function converts an array of (x,y) coordinate pairs from one coordinate space to another, using the supplied transform matrix.

Parameters

hps (HPS) – input
Presentation-space handle.

ICount (LONG) – input
Point count.
Number of coordinate pairs in *aptlPoints*.

aptlPoints (PPOINTL) – input/output
Array of (x,y) coordinate pair structures.

ICount (LONG) – input
Number of elements.

The number of elements of *pmatlfArray* to be examined, starting from the beginning of the structure. If *ICount* is less than 9, remaining elements default to the corresponding elements of the identity matrix. If *ICount* = 0, the identity matrix is used.

pmatlfArray (PMATRIXLF) – input
Instance transform matrix.

The third, sixth, and ninth elements, when specified, must be 0, 0, and 1, respectively.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE An invalid coordinate value was specified.

PMERR_INV_LENGTH_OR_COUNT An invalid length or count parameter was specified.

PMERR_COORDINATE_OVERFLOW An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

GpiConvertWithMatrix — Convert with Matrix

Remarks

The array contains x1, y1, x2, y2,... The input coordinates are replaced by the converted coordinates.

Only the supplied transform matrix is used, all other current transforms are ignored by this function.

The transform is specified as a one-dimensional array of elements, being the first *lCount* elements of a 3-row by 3-column matrix ordered by rows. The order of the elements is:

Matrix	Array
$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix}$	(a,b,0,c,d,0,e,f,1)

A point with coordinates (x,y) is transformed to the point

$(a*x + c*y + e, b*x + d*y + f)$

Example Code

This example uses GpiConvertWithMatrix to convert two coordinate pairs to another coordinate space defined by the supplied matrix, which has only the first transform element defined.

```
#define INCL_GPITRANSFORMS      /* GPI Transform functions */
#include <os2.h>

BOOL fSuccess;                /* success indicator */
HPS hps;                      /* Presentation-space handle */
LONG lCountp;                 /* Point count */
POINTL aptlPoints[2] = {{0L,0L},{1L,1L}};
/* Array of (x,y) coordinate pair
   structures */
LONG lCount;                  /* Number of elements */
MATRIXLF pmatlfArray;         /* Instance transform matrix */

lCount = 1; /* examine only first element of transform matrix */

pmatlfArray.fxM11 = 2; /* set first element of transform matrix */

fSuccess = GpiConvertWithMatrix(hps, lCountp, aptlPoints,
                                lCount, &pmatlfArray);
```

GpiCopyMetaFile – Copy Metafile

```
#define INCL_GPIMETAFILES /* Or use INCL_GPI or INCL_PM */
```

HMF GpiCopyMetaFile (HMF hmf)

This function creates a new metafile and copies the contents of an existing loaded metafile into it.

Parameters

hmf (HMF) – input
Source metafile handle.

Returns

New metafile handle and error indicators:

≠0 New metafile handle

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HMF An invalid metafile handle was specified.

PMERR_METAFILE_IN_USE An attempt has been made to access a metafile that is in use by another thread.

PMERR_TOO_MANY_METAFILES_IN_USE The maximum number of metafiles allowed for a given process was exceeded.

Remarks

The source metafile must already be loaded or generated. It is identified by a metafile handle. The new metafile is identified by a handle that is returned by this function, so it may be used, for example, by GpiPlayMetaFile.

The new metafile is owned by the process from which this function is issued. It cannot be accessed directly from any other process. If it still exists when the process terminates, it is automatically deleted by the system.

Related Functions

- GpiDeleteMetaFile
- GpiLoadMetaFile
- GpiPlayMetaFile
- GpiQueryMetaFileBits
- GpiQueryMetaFileLength
- GpiSaveMetaFile
- GpiSetMetaFileBits

GpiCopyMetaFile —

Copy Metafile

Example Code

This example uses the GpiCopyMetaFile function to make a copy of the metafile loaded using the GpiLoadMetaFile function.

```
#define INCL_GPIMETAFILES      /* Metafile functions      */
#include <os2.h>

HAB hab;                      /* anchor block handle */
HMF hmf, hmf2;                /* metafile handle      */

/* loads metafile from disk */
hmf = GpiLoadMetaFile(hab, "sample.met");
.
.
.
hmf2 = GpiCopyMetaFile(hmf);   /* copy the metafile    */
```

GpiCorrelateChain – Correlate Chain

```
#define INCL_GPICORRELATION /* Or use INCL_GPI or INCL_PM */
```

```
LONG GpiCorrelateChain (HPS hps, LONG IType, PPOINTL pptIPick, LONG IMaxHits,  
LONG IMaxDepth, PLONG alSegTag)
```

This function performs a correlate operation on the retained segment chain. It returns data for each tagged primitive that intersects the current aperture, as set by GpiSetPickApertureSize.

Parameters

hps (HPS) – input
Presentation-space handle.

IType (LONG) – input
Segment type.

Type of segment on which correlation is to be performed:

PICKSEL_VISIBLE Only visible and detectable segments with nonzero identifiers are correlated.

PICKSEL_ALL All segments with nonzero identifiers are correlated, regardless of the detectability and visibility attributes of the segments.

pptIPick (PPOINTL) – input
Pick position.

The position of the center of the pick aperture, in presentation page units.

IMaxHits (LONG) – input
Maximum hits.

Maximum number of hits that can be returned in the *alSegTag* parameter.

IMaxDepth (LONG) – input
Number of pairs.

Number of segment and tag pairs to be returned by each hit.

alSegTag (PLONG) – output
Segment identifiers and tags.

An array consisting of segment identifiers and primitive tags in alternate elements. For each hit, a set of *IMaxDepth* segment identifiers and tag pairs is returned.

Returns

Number of hits and error indicators:

≥0 Number of hits that occurred

GPI_ALTEROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE

An invalid coordinate value was specified.

PMERR_INV_MAX_HITS

An invalid maxhits parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.

GpiCorrelateChain —

Correlate Chain

PMERR_INV_CORRELATE_DEPTH	An invalid maxdepth parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.
PMERR_INV_CORRELATE_TYPE	An invalid type parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.

Remarks

The data returned for each “hit” (or correlation) consists of a set of segment and tag pairs, starting with the correlated one and followed by the one that called that segment. This is repeated until either the root segment is reached or *IMaxDepth* segment and tag pairs are returned.

Only primitives with a nonzero tag in segments with a nonzero identifier are correlated using this function. Primitives in segments called (to any depth in the hierarchy) from an unnamed segment are not eligible for correlation.

The depth value specifies the number of sets of segment and tag pairs to be returned for each hit. If the root segment is reached before *IMaxDepth* values, the remaining values are set to zero. If more than *IMaxDepth* values are available, only that number is returned.

The number of hits that occurred is returned in *INumHits*.

A “hit” is an instance of a segment identifier and tag pair for which the primitives lie completely or partially within the specified aperture. Two different primitives in the same segment might have the same tag, and would therefore produce the same hit. This is counted as a single hit; the hit is recorded only once in the *alSegTag* parameter returned. The *INumHits* parameter, therefore, returns this distinct number of hits. Hits are returned in the reverse order of their occurrence.

alSegTag is set to the hits that are found, up to the maximum defined in the *IMaxHits* parameter. Corresponding pairs of elements form the “hit” pairs. The number returned by the function therefore contains the number of sets of *IMaxDepth* pairs set if the *IMaxHits* parameter is greater than the number of hits detected. The number of elements set in the *alSegTag* parameter is twice the number returned by the function (subject to a maximum of *IMaxHits*) multiplied by the *IMaxDepth*.

If the *INumHits* value returned by the function is greater than that specified in *IMaxHits*, more hits occurred than could be returned. If all hits are important, specify an array that is large enough to contain the maximum number of sets of hits that are expected.

The draw controls (see GpiSetDrawControl) are ignored by this function.

It may be necessary to ensure that attributes, model transform, current position, and viewing limits are reset to their default values, before processing the chain. This can be done by either ensuring that the first segment to be correlated does not have the ATTR_FASTCHAIN attribute (see GpiSetInitialSegmentAttrs), or by issuing GpiResetPS before the GpiCorrelateChain. The latter method also resets the clip path to no clipping.

If this function is followed by primitives or attributes, without first opening a segment, the processing is as described for GpiCloseSegment.

GpiCorrelateChain – Correlate Chain

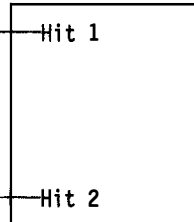
Examples

Start segment 1
Tag 10
Call 2
End segment 1

Start segment 2
Tag 20
Call 3
Tag 21
.....
.....
End segment 2

Start segment 3
Tag 30
.....
.....
End segment 3

Pick Aperture



For *IMaxHits* = 1 at *IMaxDepth* = 2:

segment	tag
2	21
1	10

Returned *INumHits* = 2.

For *IMaxHits* = 2 at *IMaxDepth* = 4:

segment	tag	
2	21	hit1.1
1	10	hit1.2
0	0	hit1.3
0	0	hit1.4
3	30	hit2.1
2	20	hit2.2
1	10	hit2.3
0	0	hit2.4

Returned *INumHits* = 2.

Related Functions

- GpiCorrelateFrom
- GpiCorrelateSegment
- GpiSetDrawControl
- GpiSetPickAperturePosition
- GpiSetPickApertureSize

GpiCorrelateChain —

Correlate Chain

Example Code

This example uses GpiCorrelateChain to correlate, using an aperture of default size and centered at (200,200), on visible and detectable segments and requests one intersection (or hit) and one segment/tag pair for that hit to be returned. The segments will have been previously defined and created using GpiSetInitialSegmentAttrs and GpiOpenSegment/GpiCloseSegment.

```
#define INCL_GPICORRELATION    /* GPI Correlation functions */
#include <os2.h>

BOOL    fSuccess;           /* success indicator */
SIZEL   pszlSize={0L,0L}; /* size of pick aperture */
LONG    lNumHits;           /* number of hits or error */
HPS     hps;               /* Presentation-space handle */
POINTL   pptlPick = {200L,200L};
                        /* Pick (center of aperture) position */
LONG     lMaxHits;          /* Maximum hits to be returned */
LONG     lMaxDepth;         /* Number of pairs to be returned */
LONG     alSegTag;          /* Segment identifiers and tags */

fSuccess = GpiSetPickAperturePosition(hps, &pptlPick);

/* set aperture size (use default) */
fSuccess = GpiSetPickApertureSize(hps, PICKAP_DEFAULT, &pszlSize);

/* return only one hit */
lMaxHits = 1L;

/* return only one segment/tag pair per hit */
lMaxDepth = 1L;

/* correlate on visible, detectable segment chains */
lNumHits = GpiCorrelateChain(hps, PICKSEL_VISIBLE, &pptlPick, lMaxHits,
                             lMaxDepth, &alSegTag);
```

GpiCorrelateFrom – Correlate From

```
#define INCL_GPICORRELATION /* Or use INCL_GPI or INCL_PM */
```

```
LONG GpiCorrelateFrom (HPS hps, LONG IFirstSegment, LONG ILastSegment, LONG IType,  
PPOINTL pptlPick, LONG IMaxHits, LONG IMaxDepth,  
PLONG alSegTag)
```

This function performs a correlate operation on a section of the retained segment chain.

Parameters

hps (HPS) – input

Presentation-space handle.

IFirstSegment (LONG) – input

Specifies the first segment to be correlated.

It must be greater than 0.

ILastSegment (LONG) – input

Specifies the last segment to be correlated.

It must be greater than 0.

IType (LONG) – input

Type of segments on which correlation is to be performed:

PICKSEL_VISIBLE Only visible and detectable segments with nonzero identifiers are correlated.

PICKSEL_ALL All segments with nonzero identifiers are correlated, regardless of the detectability and visibility attributes of the segments.

pptlPick (PPOINTL) – input

Pick position.

The position of the center of the pick aperture, in presentation page units.

IMaxHits (LONG) – input

Maximum hits.

Maximum number of hits that can be returned in the *alSegTag* parameter.

IMaxDepth (LONG) – input

Number of pairs.

Number of segment and tag pairs to be returned by each hit.

alSegTag (PLONG) – output

Segment identifiers and tags.

An array consisting of segment identifiers and primitive tags in alternate elements. For each hit, a set of *IMaxDepth* segment identifiers and tag pairs is returned.

GpiCorrelateFrom — Correlate From

Returns

Number of hits and error indicators:

≥ 0 Number of hits that occurred

GPI_ALTEERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_CORRELATE_TYPE	An invalid type parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.
PMERR_INV_MAX_HITS	An invalid maxhits parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.
PMERR_INV_CORRELATE_DEPTH	An invalid maxdepth parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.
PMERR_SEG_NOT_FOUND	The specified segment identifier did not exist
PMERR_SEG_NOT_CHAINED	An attempt was made to issue GpiDrawFrom, GpiCorrelateFrom or GpiQuerySegmentPriority for a segment that was not chained.
PMERR_INV_SEG_NAME	An invalid segment identifier was specified.

Remarks

The correlation operation starts at the segment identified by *IFirstSegment* and includes chained and called segments up to, and including, the segment identified by *ILastSegment*.

Data is returned for each tagged primitive that intersects the pick aperture. The data returned for each "hit" (or correlation) consists of a set of segment and tag pairs, starting with the correlated one and followed by the one that called the segment. This is repeated until the root segment is reached or *IMaxDepth* values are returned.

Only primitives with a nonzero tag (see GpiSetTag) in segments with a nonzero identifier are correlated using this function. Primitives in segments called (to any depth in the hierarchy) from a segment 0 are not eligible for correlation.

The depth value specifies the number of sets of segment and tag pairs to be returned for each hit. If the root segment is reached before *IMaxDepth* values, the remaining values are set to zero. If more than *IMaxDepth* values are available, only that number is returned.

The number of hits that occurred is returned in *INumHits*.

A "hit" is an instance of a segment identifier and tag pair for which the primitives lie completely or partially within the specified aperture. Two different primitives in the same segment might have the same tag, and would therefore produce the same hit. This is counted as a single hit; the hit is recorded only once in the *aISegTag* parameter returned. The *INumHits* parameter, therefore, returns this distinct number of hits. Hits are returned in reverse order of their occurrence.

GpiCorrelateFrom — Correlate From

aISegTag is set to the hits that are found, up to the maximum defined in the *IMaxHits* parameter. Corresponding pairs of elements form the hit pairs. The number returned by the call therefore contains the number of sets of *IMaxDepth* pairs set if the *IMaxHits* parameter is greater than the number of hits detected. The number of elements set in the *aISegTag* parameter is twice the number returned by the function (subject to a maximum of *IMaxHits*) multiplied by the *IMaxDepth*.

If the *INumHits* value returned by the function is greater than that specified in *IMaxHits*, more hits occurred than could be returned. If all hits are important, specify an array that is large enough to contain the maximum number of sets of hits that are expected.

The draw controls (see `GpiSetDrawControl`) are ignored by this call.

It may be necessary to ensure that attributes, model transform, current position, and viewing limits are reset to their default values, before processing the segments. This can be done either ensuring that the first segment to be correlated does not have the `ATTR_FASTCHAIN` attribute (see `GpiSetInitialSegmentAttrs`), or by issuing `GpiResetPS` before the `GpiCorrelateFrom`. The latter method also resets the clip path to no clipping.

If this function is followed by primitives or attributes, without first opening a segment, the processing is as described for `GpiCloseSegment`.

If *IFirstSegment* does not exist, or is not in the segment chain, an error is raised. If *ILastSegment* does not exist, or is not in the chain, or is chained before *IFirstSegment*, no error is raised and processing continues to the end of the chain.

Related Functions

- `GpiCorrelateChain`
- `GpiCorrelateSegment`
- `GpiSetDrawControl`
- `GpiSetPickAperturePosition`
- `GpiSetPickApertureSize`

GpiCorrelateFrom — Correlate From

Example Code

This example uses GpiCorrelateFrom to correlate, using an aperture of default size and centered at (200,200), on visible and detectable segments within the given chain of 2 segments. It requests one intersection (or hit) and one segment/tag pair for that hit to be returned. The segments will have been previously defined and created using GpiSetInitialSegmentAttrs and GpiOpenSegment/GpiCloseSegment.

```
#define INCL_GPICORRELATION    /* GPI Correlation functions */
#include <os2.h>

BOOL    fSuccess;           /* success indicator */
SIZEL   pszlSize;           /* size of pick aperture */
LONG    lNumHits;           /* number of hits or error */
HPS     hps;                /* Presentation-space handle */
LONG    lFirstSegment;      /* Specifies the first segment to be
                             correlated */
LONG    lLastSegment;       /* Specifies the last segment to be
                             correlated */
POINTL   pptlPick = {200L,200L};
                             /* Pick (center of aperture) position */
LONG     lMaxHits;          /* Maximum hits to be returned */
LONG     lMaxDepth;         /* Number of pairs to be returned */
LONG     alSegTag;          /* Segment identifiers and tags */

fSuccess = GpiSetPickAperturePosition(hps, &pptlPick);

/* set aperture size (use default) */
fSuccess = GpiSetPickApertureSize(hps, PICKAP_DEFAULT, &pszlSize);

/* define chain of two segments (1 and 2) */
lFirstSegment = 1;
lLastSegment = 2;

/* return only one hit */
lMaxHits = 1L;

/* return only one segment/tag pair per hit */
lMaxDepth = 1L;

/* correlate on visible, detectable segments */
lNumHits = GpiCorrelateFrom(hps, lFirstSegment, lLastSegment,
                           PICKSEL_VISIBLE, &pptlPick, lMaxHits,
                           lMaxDepth, &alSegTag);
```

GpiCorrelateSegment – Correlate Segment

```
#define INCL_GPICORRELATION /* Or use INCL_GPI or INCL_PM */
```

LONG GpiCorrelateSegment (HPS *hps*, LONG *ISegment*, LONG *IType*, PPOINTL *pptIPick*,
LONG *IMaxHits*, LONG *IMaxDepth*, PLONG *alSegTag*)

This function performs a correlate operation on a specified segment.

Parameters

hps (HPS) – input

Presentation-space handle.

ISegment (LONG) – input

Identifier of the segment to be correlated.

It must be greater than 0.

IType (LONG) – input

Type of segments on which correlation is to be performed:

PICKSEL_VISIBLE Only visible and detectable segments with nonzero identifiers are correlated.

PICKSEL_ALL All segments with nonzero identifiers are correlated, regardless of the detectability and visibility attributes of the segments.

pptIPick (PPOINTL) – input

Pick position.

The position of the center of the pick aperture, in presentation page units.

IMaxHits (LONG) – input

Maximum hits.

The maximum number of hits that can be returned in the *alSegTag* parameter.

IMaxDepth (LONG) – input

Number of pairs.

Number of segment/tag pairs to be returned by each hit.

alSegTag (PLONG) – output

Segment identifiers and tags.

An array consisting of segment identifiers and primitive tags in alternate elements. For each hit, a set of *IMaxDepth* segment identifiers and tag pairs is returned.

Returns

Number of hits and error indicators:

≥0 Number of hits that occurred

GPI_ALTERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_CORRELATE_TYPE

An invalid type parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.

GpiCorrelateSegment — Correlate Segment

PMERR_INV_COORDINATE	An invalid coordinate value was specified.
PMERR_INV_MAX_HITS	An invalid maxhits parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.
PMERR_INV_CORRELATE_DEPTH	An invalid maxdepth parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.
PMERR_SEG_NOT_FOUND	The specified segment identifier did not exist
PMERR_INV_SEG_NAME	An invalid segment identifier was specified.

Remarks

Data is returned for each tagged primitive that intersects the pick aperture. The data returned for each "hit" (or correlation) consists of a set of segment and tag pairs, starting with the correlated one and followed by the one that called that segment. This is repeated until the specified segment (which was not called by another segment) is reached, or *IMaxDepth* values are returned.

The specified segment identifier must be nonzero. Only primitives with a nonzero tag (see GpiSetTag) are correlated using this function.

The depth value specifies the number of sets of segment and tag pairs to be returned for each hit. If the specified segment is reached before *IMaxDepth* values, the remaining values are set to zero. If more than *IMaxDepth* values are available, only that number is returned.

The number of hits that occurred is returned in *INumHits*.

A "hit" is an instance of a segment identifier and tag pair for which the primitives lie completely or partially within the specified aperture. Two different primitives in the same segment might have the same tag, and would therefore produce the same hit. This is counted as a single hit; the hit is recorded only once in the *aISegTag* parameter returned. The *INumHits* parameter, therefore, returns this distinct number of hits. Hits are returned in reverse order of their occurrence.

aISegTag is set to the hits that are found, up to the maximum defined in the *IMaxHits* parameter. Corresponding pairs of elements form the hit pairs. The number returned by the function, therefore, contains the number of sets of *IMaxDepth* pairs set if the *IMaxHits* parameter is greater than the number of hits detected. The number of elements set in the *aISegTag* parameter is twice the number returned by the function (subject to a maximum of *IMaxHits*) multiplied by the *IMaxDepth*.

If the *INumHits* value returned by the function is greater than that specified in *IMaxHits*, more hits occurred than could be returned. If all hits are important, specify an array that is large enough to contain the maximum number of sets of hits that are expected.

The draw controls (see GpiSetDrawControl) are ignored by this function. This function differs from the other GpiCorrelate... functions because the segment to be correlated need not be a chained segment.

It may be necessary to ensure that attributes, model transform, current position, and viewing limits are reset to their default values before processing the segment. This can be done either by ensuring that the segment to be correlated does not have the ATTR_FASTCHAIN attribute (see GpiSetInitialSegmentAttrs) or by issuing GpiResetPS before the GpiCorrelateSegment. The latter method also resets the clip path to no clipping.

If this function is followed by primitives or attributes without first opening a segment, the processing is as described for GpiCloseSegment.

GpiCorrelateSegment – Correlate Segment

Related Functions

- GpiCorrelateChain
- GpiCorrelateFrom
- GpiCallSegmentMatrix
- GpiCloseSegment
- GpiDeleteSegment
- GpiDeleteSegments
- GpiDrawSegment
- GpiErrorSegmentData
- GpiOpenSegment
- GpiQueryInitialSegmentAttrs
- GpiQuerySegmentAttrs
- GpiQuerySegmentNames
- GpiQuerySegmentPriority
- GpiSetDrawControl
- GpiSetInitialSegmentAttrs
- GpiSetPickAperturePosition
- GpiSetPickApertureSize
- GpiSetSegmentAttrs
- GpiSetSegmentPriority

GpiCorrelateSegment —

Correlate Segment

Example Code

This example uses GpiCorrelateSegment to correlate, using an aperture of default size and centered at (200,200), on a visible and detectable segment and requests one intersection (or hit) and one segment/tag pair for that hit to be returned. The segment will have been previously defined and created using GpiSetInitialSegmentAttrs and GpiOpenSegment/GpiCloseSegment.

```
#define INCL_GPICORRELATION    /* GPI Correlation functions */
#include <os2.h>

BOOL    fSuccess;           /* success indicator */
SIZEL   pszlSize;           /* size of pick aperture */
LONG    lNumHits;           /* number of hits or error */
HPS     hps;                /* Presentation-space handle */
LONG    lSegment;           /* segment to be correlated */
LONG    lLastSegment;       /* Specifies the last segment to be
                             correlated */
POINTL   pptlPick = {200L,200L};
                             /* Pick (center of aperture) position */
LONG    lMaxHits;           /* Maximum hits to be returned */
LONG    lMaxDepth;          /* Number of pairs to be returned */
LONG    alSegTag;           /* Segment identifiers and tags */

fSuccess = GpiSetPickAperturePosition(hps, &pptlPick);

/* set aperture size (use default) */
fSuccess = GpiSetPickApertureSize(hps, PICKAP_DEFAULT, &pszlSize);

/* define segment */
lSegment = 1;

/* return only one hit */
lMaxHits = 1L;

/* return only one segment/tag pair per hit */
lMaxDepth = 1L;

/* correlate on visible, detectable segments */
lNumHits = GpiCorrelateSegment(hps, lSegment, PICKSEL_VISIBLE,
                               &pptlPick, lMaxHits, lMaxDepth,
                               &alSegTag);
```

GpiCreateBitmap – Create Bit Map

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI or INCL_PM */
```

HBITMAP GpiCreateBitmap (HPS hps, PBITMAPINFOHEADER2 pbmp2New, ULONG fOptions, PBYTE pblnitData, PBITMAPINFO2 pbmi2InfoTable)

This function creates a bit map and returns the bit-map handle.

Parameters

hps (HPS) – input

Presentation-space handle.

The associated device should, if possible, hold the bit map in its own memory. Where this is not possible, main memory is used and the bit map is held in a format compatible with the device.

pbmp2New (PBITMAPINFOHEADER2) – input

Bit-map information header.

This structure defines the format of the bit map to be created.

fOptions (ULONG) – input

Options:

CBM_INIT Initialize the bit map with *pblnitData*

If the bit map is stored on a device, the *fOptions* parameter is passed to the device. Bits 16 through 31 can be used for special features known to be supported by the particular device driver.

pblnitData (PBYTE) – input

Buffer address.

The address in application storage from which initialization data is to be copied, if CBM_INIT is set.

pbmi2InfoTable (PBITMAPINFO2) – input

Bit-map information table.

This defines the format of the data in *pblnitData*. It is ignored if CBM_INIT is not set.

Returns

Bit-map handle and error indicators:

≠0 New bit-map handle

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_INFO_TABLE

An invalid bit-map info table was specified with a bit-map operation.

PMERR_INV_USAGE_PARM

An invalid options parameter was specified with GpiCreateBitmap.

GpiCreateBitmap —

Create Bit Map

Remarks

On some devices it is possible to create the bit map in device memory. Even when this is not possible, a bit map always belongs to a particular device. The device is specified through the device context associated with the specified presentation space. The device context can be any device context that describes the physical device (such as any window device context for the screen).

There are a number of standard bit-map formats that should normally be adhered to. Other formats can be used if supported by the device.

A newly created bit map can be filled with data supplied by the application. This is useful where the bit map always contains, or always starts with, the same image, captured in the application. A bit-map information structure is also passed, which defines the format and color usage of the initialization data. It is assumed that enough data is passed to initialize the entire bit map.

Some bit-map functions, including those that draw into the bit map, require the bit map to be selected into a memory device context, using `GpiSetBitmap`. This is true whether device or main memory is used to hold the bit map.

The bit map is owned by the process from which this function is issued. It cannot be accessed directly from any other process. If it still exists when the process terminates, it is automatically deleted by the system.

Some restrictions apply when using this function. Refer to Appendix G, "Format of Interchange Files" on page G-1 for additional details.

Related Functions

- `GpiBitBlt`
- `GpiDeleteBitmap`
- `GpiDrawBits`
- `GpiLoadBitmap`
- `GpiQueryBitmapBits`
- `GpiQueryBitmapDimension`
- `GpiQueryBitmapHandle`
- `GpiQueryBitmapParameters`
- `GpiQueryDeviceBitmapFormats`
- `GpiSetBitmap`
- `GpiSetBitmapBits`
- `GpiSetBitmapDimension`
- `GpiSetBitmapId`
- `GpiWCBitBlt`
- `WinDrawBitmap`
- `WinGetSysBitmap`

GpiCreateBitmap – Create Bit Map

Example Code

The following example loads a bit map resource from memory and uses the GpiCreateBitmap function to create the bit map. This is similar to using the GpiLoadBitmap function, except it gives the application the chance to modify the bit map image data before creating the bit map.

```
#define INCL_GPIBITMAPS      /* GPI bit map functions      */
#define INCL_DOSRESOURCES    /* Dos Resource functions */
#include <os2.h>

HPS hps;                    /* presentation space handle */
                             /* address of bit map image data in */
                             resource /*
BITMAPINFOHEADER2 bmih;    /* bit map info structure      */
HBITMAP hbm;               /* bit map handle              */

memset (&bmih,0, sizeof(BITMAPINFOHEADER2));
bmih.cbFix      = sizeof(BITMAPINFOHEADER2);
bmih.cx         = cx;
bmih.cy         = cy;
bmih.cPlanes    = 1;
bmih.cBitCount  = cBitCount;
(hbm = GpiCreateBitmap(hps, &bmih, 0L, NULL, NULL));
```

GpiCreateLogColorTable – Create Logical Color Table

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiCreateLogColorTable (HPS hps, ULONG flOptions, LONG IFormat, LONG IStart, LONG ICount, PLONG aiTable)

This function defines the entries of the logical color table.

Parameters

hps (HPS) – input
Presentation-space handle.

flOptions (ULONG) – input
Options:

LCOL_RESET The color table is reset to its default values before processing the remainder of the data in this function.

This value is assumed if the color table is currently in RGB mode and is being changed to index mode; that is, LCOLF_INDRGB or LCOLF_CONSECRGB is specified.

The *IFormat* parameter must be LCOLF_INDRGB or LCOLF_CONSECRGB.

LCOL_PURECOLOR When this option is set only colors for solid patterns (see GpiSetPattern) available in the physical palette will be used. Only pure colors are used and no dithering is done.

Other flags are reserved and must be 0.

IFormat (LONG) – input
Format of entries in the table:

LCOLF_INDRGB Array of index/RGB pairs. Each pair is 8 bytes long: 4 bytes (local format) for the index, and 4 bytes for the color value.

This sets the color table into index mode (and forces LCOL_RESET) if it is in RGB mode.

The maximum index that can be loaded is returned in the CAPS_COLOR_INDEX parameter of the DevQueryCaps function.

Each index specified must be greater than or equal to 0.

LCOLF_CONSECRGB Array of RGB values, corresponding to color indexes *IStart* upwards. Each entry is 4 bytes long.

This sets the color table into index mode (and forces LCOL_RESET) if it is in RGB mode.

The maximum index that can be loaded is returned in the CAPS_COLOR_INDEX parameter of the DevQueryCaps function.

LCOLF_RGB Color index = RGB.

This sets the color table into RGB mode.

IStart (LONG) – input
Starting index.

This is relevant only for LCOLF_CONSECRGB.

The starting index must be greater than or equal to 0.

GpiCreateLogColorTable – Create Logical Color Table

lCount (LONG) – input

Count of elements in *aTable*.

This must be greater than or equal to 0. If 0 is specified, LCOLF_INDRGB and LCOLF_CONSECRGB have the same effect.

For LCOLF_INDRGB, *aTable* must contain an even number of elements. *lCount* must be an even number.

aTable (PLONG) – input

Start of the application data area.

This contains the color table definition data. The format depends on the value of *lFormat*.

Each color value is a 4-byte integer, with a value of

$$(R * 65536) + (G * 256) + B$$

where:

R is red intensity value

G is green intensity value

B is blue intensity value.

The maximum intensity for each primary is 255.

The high order byte must be 0.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COLOR_OPTIONS

An invalid options parameter was specified with a logical color table or color query function.

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

PMERR_INV_COLOR_DATA

Invalid color table definition data was specified with GpiCreateLogColorTable.

PMERR_INV_COLOR_FORMAT

An invalid format parameter was specified with GpiCreateLogColorTable.

PMERR_INV_COLOR_START_INDEX

An invalid starting index parameter was specified with a logical color table or color query function.

PMERR_REALIZE_NOT_SUPPORTED

An attempt was made to create a realizable logical color table on a device driver that does not support this function.

PMERR_PALETTE_SELECTED

Color palette operations cannot be performed on a presentation space while a palette is selected.

GpiCreateLogColorTable — Create Logical Color Table

Remarks

This function can cause the color table to be reset to the default values. These are:

CLR_BACKGROUND	Reset color, used by GpiErase. This is the natural background color for the device. For a display, it is the default window color (SYSCLR_WINDOWTEXT; see WinSetSysColors). For a printer, it is the paper color. The background color for the display can be changed by setting new system colors from the Control Panel. The background color for a printer can be changed by selecting a new paper color (if allowed by the presentation driver).
CLR_BLUE	Blue.
CLR_RED	Red.
CLR_PINK	Pink (magenta).
CLR_GREEN	Green.
CLR_CYAN	Cyan (turquoise).
CLR_YELLOW	Yellow.
CLR_NEUTRAL	A device-dependent color that provides a contrasting color to CLR_BACKGROUND. For a display, it is the default window text color (SYSCLR_WINDOWTEXT; see WinSetSysColors). For a printer, it is a color that contrasts with the paper color. The neutral color for the display can be changed by setting new system colors from the Control Panel. The neutral color for a printer can be changed by selecting a new paper color (if allowed by the presentation driver).
CLR_DARKGRAY	Dark gray.
CLR_DARKBLUE	Dark blue.
CLR_DARKRED	Dark red.
CLR_DARKPINK	Dark pink.
CLR_DARKGREEN	Dark green.
CLR_DARKCYAN	Dark cyan.
CLR_BROWN	Brown.
CLR_PALEGRAY	Pale gray.

GpiErase clears the output of a device to the color defined by CLR_BACKGROUND.

By default, presentation spaces have a logical color table consisting of the 16 default values given above. In index mode, these entries are always considered as part of the color table, unless they are explicitly overwritten. Color indexes outside this range, which have not been loaded, are not considered as part of the color table; it is an error to use such colors if the color table is in index mode.

The system performs a mapping from the colors in the logical color table to those in the standard physical color table for that device. This mapping is used for all drawing and bit maps. Mixing is not predictable.

The standard physical color table always includes the standard 16 colors, where this is physically possible. On devices that support more than 16 colors, there may be additional colors available to which the requested colors may be mapped. However, it cannot be ensured that these additional colors are the same on different devices. Applications that depend upon precise colors beyond the first 16 should use a palette (see GpiCreatePalette) on devices for which this is supported. DevQueryCaps can be used to determine whether the function is supported by the device; see CAPS_PALETTE_MANAGER.

For a monochrome device (whether it is a display, bit map, printer, or some other type), a reset color is defined as follows:

1. Start with the appropriate item below:
 - The paper color, for a printer with no loaded color table

GpiCreateLogColorTable – Create Logical Color Table

- SYSCLR_WINDOW, for a monochrome display with no loaded color table
 - Color 0, for any device if a color table has been loaded.
2. If this color is white or a light color, the reset color is set to white; otherwise, the reset color is set to black.

The reset color is used for:

- The color that GpiErase clears the output to
- CLR_BACKGROUND (color 0), unless an RGB color table is in use
- CLR_DEFAULT for GpiSetBackColor
- Any color that has exactly the same RGB value as the reset color.

Any other color becomes black if the reset color is white, and the converse.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see “Metafile Restrictions” on page G-1.

Related Functions

- DevQueryCaps
- GpiCreatePalette
- GpiQueryColorData
- GpiQueryColorIndex
- GpiQueryLogColorTable
- GpiQueryNearestColor
- GpiQueryRealColors
- GpiQueryRGBColor
- WinSetSysColors

Example Code

This example uses the GpiCreateLogColorTable function to create a logical color table, using data from the previous logical color table.

```
#define INCL_GPILOGCOLORTABLE /* Color Table functions */
#include <os2.h>

HPS hps; /* presentation space handle */
LONG a1Table[16]; /* assume 16 entries */

/* retrieve the current table */

GpiQueryLogColorTable(hps, 0L, 0L, 16L, a1Table);

a1Table[1] = 0x000080; /* change the second entry to light blue */

GpiCreateLogColorTable(hps, /* presentation space */
    0L, /* no special options */
    LCOLF_CONSECRGB, /* consecutive RGB values */
    0L, /* start with color index 0 */
    16, /* 16 entries */
    a1Table); /* RGB color values */
```

GpiCreateLogFont — Create Logical Font

```
#define INCL_GPILCIDS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiCreateLogFont (HPS hps, PSTR8 pName, LONG lLcid, PFATTRS pAttrs)

This function provides a logical definition of a font.

Parameters

hps (HPS) — input
Presentation-space handle.

pName (PSTR8) — input
Logical font name.

An 8-character name that can be used to describe the logical font. Its principal use is in interchange files, where it can help to identify the required font. For example, it can reference a file name that contains the font for a remote system.

lLcid (LONG) — input
Local identifier.

The local identifier that the application uses to refer to this font. It must be in the range 0 through 254. If 0 is specified, the properties of the default font are changed. The original default font can be restored by calling GpiDeleteSetId, with an *lLcid* parameter of LCID_DEFAULT or LCID_ALL.

If the *lLcid* parameter specifies a local identifier that is already being used to refer to a logical font, but is not the current pattern-set or marker-set local identifier, then the new definition replaces the old one. If *lLcid* specifies a local identifier that is already being used to refer to a logical font, and is the current pattern-set or marker-set local identifier, an error occurs. An error also occurs if the local identifier is currently used to refer to a bit map.

pAttrs (PFATTRS) — input
Attributes required of the font.

Returns

Match indicators:

FONT_MATCH	Font requirements matched successfully
FONT_DEFAULT	Font requirements not matched; a default font is used
GPI_ERROR	Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_SETID	An invalid setid parameter was specified.
PMERR_INV_FONT_ATTRS	An invalid attrs parameter was specified with GpiCreateLogFont.
PMERR_FONT_NOT_LOADED	An attempt was made to create a font that was not loaded.
PMERR_SETID_IN_USE	An attempt was made to specify a setid that was already in use as the currently selected character, marker or pattern set.

GpiCreateLogFont — Create Logical Font

PMERR_KERNING_NOT_SUPPORTED

Kerning was requested on GpiCreateLogFont call to a presentation space associated with a device context that does not support kerning.

Remarks

The system uses the available physical font that most closely matches the requirements. Physical fonts can be:

- Loaded at initialization time
- Built into particular devices or device drivers
- Private ones for this process, loaded by GpiLoadFonts.

An application can force selection of a particular physical font by quoting the *IMatch* value in *FATTRS* to be returned for the desired font by GpiQueryFonts. However, this method is only valid for a particular device/device driver combination on a single machine. This method should be avoided as a method for selecting fonts.

Whichever method is used, the choice of physical font, which is made when this function is issued, is never subsequently changed for a particular logical font.

The local identifier (*lLcid*) that the application decides to use to reference this logical font for later drawing operations is also specified; see GpiSetCharSet.

If the face name is provided, GpiCreateLogFont tries to select the font with that face name. If the face name is empty, GpiCreateLogFont selects a default font.

When a match number is provided, GpiCreateLogFont tries to find a font with the same match number and face name. If there is a mismatch at this point, GpiCreateLogFont acts as though the match number is 0 and starts the search again.

When the match number is 0 and the calling program requests a bit-map font (*FATTR_FONTUSE_OUTLINE* not set), GpiCreateLogFont searches for a bit-map font with the required average character width (*AveCharWidth*) and maximum baseline extent (*MaxBaselineExt*), consistent with the usage flags. If this search fails, GpiCreateLogFont searches for an outline font with the required face name.

When the match number is zero and the calling program requests an outline font (*FATTR_FONTUSE_OUTLINE* is set), GpiCreateLogFont searches for an outline font with the required selection flags. If that search fails, a default outline font is selected. If the match number is set to a positive number, a Presentation Manager font is selected. If the match number is negative, a font belonging to a physical device is selected.

It is advisable to set the values of *all* the elements in the *pAttr*s structure. This is particularly important where printing, plotting, or interchange are concerned, as the target machine may need to substitute an existing device font for the requested font.

To anticipate possible substitution by a vector font, values should be set for character angle, character shear and character box (using GpiSetCharAngle, GpiSetCharShear, and GpiSetCharBox respectively) before drawing any character strings. The GpiQueryFontMetrics function can be used to get the values of the character box height and width for a font. These are held in the fields *lEmHeight* and *lEmInc* in the *FONTMETRICS* structure.

Outline font characters are normally drawn filled. However, hollow characters are produced if the *FATTR_SEL_OUTLINE* flag is set in the *pAttr*s parameter. For small characters, outlining in this way can give a similar visual appearance to filled characters, with improved performance.

There are restrictions on the use of non-installed fonts with certain device types. See GpiLoadFonts for more details.

GpiCreateLogFont —

Create Logical Font

If this function occurs within a path definition when the drawing mode (see GpiSetDrawingMode) is **retain** or **draw-and-retain**, its effect is not stored with the definition.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see “Metafile Restrictions” on page G-1.

Related Functions

- GpiDeleteSetId
- GpiLoadFonts
- GpiQueryFontMetrics
- GpiQueryFonts
- GpiQueryKerningPairs
- GpiQueryNumberSetIds
- GpiQuerySetIds
- GpiQueryWidthTable
- GpiSetCharSet
- GpiSetCharMode
- GpiSetMarkerSet
- GpiSetPatternSet
- GpiUnloadFonts

Example Code

This example uses the GpiCreateLogFont function to create a logical font with the local identifier 1. The logical font has the face name “Courier” and requested width and height of 12 pels. Once the font is created, the example sets the font using the local identifier and displays a string in the font at the point (100,100).

```
#define INCL_GPILCIDS           /* Font functions           */
#define INCL_GPIPRIMITIVES     /* GPI primitive functions */
#include <os2.h>

HPS hps;           /* presentation space handle */
POINTL ptl = { 100, 100 };
FATTRS fat;

fat.usRecordLength = sizeof(FATTRS); /* sets size of structure */
fat.fsSelection = 0;                /* uses default selection */
fat.lMatch = 0L;                    /* does not force match */
fat.idRegistry = 0;                /* uses default registry */
fat.usCodePage = 850;              /* code-page 850 */
fat.lMaxBaselineExt = 12L;         /* requested font height is 12 pels */
fat.lAveCharWidth = 12L;          /* requested font width is 12 pels */
fat.fsType = 0;                   /* uses default type */
fat.fsFontUse = FATTR_FONTUSE_NOMIX; /* doesn't mix with graphics */

/* Copy Courier to szFacename field */
strcpy(fat.szFacename, "Courier");

GpiCreateLogFont(hps,           /* presentation space */
                NULL,          /* does not use logical font name */
                1L,            /* local identifier */
                &fat);         /* structure with font attributes */

GpiSetCharSet(hps, 1L);        /* sets font for presentation space */
GpiCharStringAt(hps, &ptl, 5L, "Hello"); /* displays a string */
```

GpiCreatePalette – Create Palette

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI or INCL_PM */
```

HPAL GpiCreatePalette (HAB hab, ULONG flOptions, LONG IFormat, LONG ICount, PLONG aiTable)

This function creates and initializes a color palette.

Parameters

hab (HAB) – input
Anchor-block handle.

flOptions (ULONG) – input
Options:

LCOL_PURECOLOR

The application does not want color *dithering* to create colors not available in the physical palette for solid patterns (see GpiSetPattern). If this option is set, only pure colors are used and no dithering is done.

LCOL_OVERRIDE_DEFAULT_COLORS

Override option for applications that need the full hardware palette. The system does not guarantee a consistent look to the user interface when this option is used. The override is only in effect while the overriding palette is in the foreground

To combine these two options, OR the values together. Other flags are reserved and must be B'0'.

IFormat (LONG) – input
Format of entries in the table:

LCOLF_CONSECRGB Array of (RGB) values. Each entry is 4 bytes long. This is currently the only supported value for this parameter.

ICount (LONG) – input
Count of elements in *aiTable*.

This must be greater than zero.

aiTable (PLONG) – input
Start of the application data area.

This contains the palette definition data.

Each color value is a 4-byte integer, with a value of

$(F * 16777216) + (R * 65536) + (G * 256) + B$

where:

F is a flag byte, which can take the following values (these can be ORed together if required):

PC_RESERVED This index is an animating index. This means that the application might frequently change the RGB value and so the system should not map the logical index of another application's palette to the entry in the physical palette used for this color.

PC_EXPLICIT The low-order word of the logical color table entry designates a physical palette slot from which the color definition is to be taken. This allows an application to show the actual contents of the device palette as realized for other logical palettes. This does not prevent the color in the slot from being changed for any reason.

GpiCreatePalette —

Create Palette

R is red intensity value
G is green intensity value
B is blue intensity value.

Each intensity value must be in the range 0 through 255.

Returns

Palette handle:

≠0 Palette handle

GPI_ERROR Error occurred.

Possible returns from WinGetLastError

PMERR_INV_COLOR_OPTIONS	An invalid options parameter was specified with a logical color table or color query function.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_INV_COLOR_DATA	Invalid color table definition data was specified with GpiCreateLogColorTable.
PMERR_INV_COLOR_FORMAT	An invalid format parameter was specified with GpiCreateLogColorTable.
PMERR_INV_COLOR_START_INDEX	An invalid starting index parameter was specified with a logical color table or color query function.
PMERR_INSUFFICIENT_MEMORY	The operation terminated through insufficient memory.

Remarks

The new palette contains only the entries set in the *a/Table* parameter. All color indices outside this range are not considered part of the palette; it is an error to use such colors when this palette is selected.

When a palette is realized (see WinRealizePalette), the lowest indices are considered first. The palette should therefore be ordered so that the most important colors have the lowest indices. Animating indices, which on realization can have their own individual slots in the physical palette, should be used only when necessary.

Palettes should be created with only those color indices that the application requires and not unnecessarily create a large palette. The maximum index for a palette is not limited to CAPS_COLOR_INDEX.

The palette can be selected into a presentation space using GpiSelectPalette.

Related Functions

- DevQueryCaps
- GpiAnimatePalette
- GpiDeletePalette
- GpiQueryPalette
- GpiQueryPaletteInfo
- GpiSelectPalette
- GpiSetPaletteEntries
- WinRealizePalette
- GpiCreateLogColorTable

GpiCreatePalette – Create Palette

Example Code

The uses GpiCreatePalette to create and initialize a palette of 4 pure (no dithering) colors.

```
#define INCL_GPILOGCOLORTABLE  /* Color Table functions      */
#include <os2.h>

HAB hab;          /* anchor block handle      */
HPAL hpal;        /* palette handle          */
LONG lFormat;     /* table entry format      */

/*****
 * assume 4 entries in palette.
 * The RGB values are calculated with the following formula:
 *   (F * 16777216) + (R * 65536) + (G * 256) + B
 *   where F = flag, PC_RESERVED or PC_EXPLICIT
 *           R = red intensity value
 *           G = green intensity value
 *           B = blue intensity value
 * Thus, in the following table, red and green intensities are 0
 * while the blue intensity increases from 1 to 4.
 *****/

ULONG aulTable[4]=
    {(PC_RESERVED*16777216) + (0*65536) + (0*256) + 1,
     (PC_RESERVED*16777216) + (0*65536) + (0*256) + 2,
     (PC_RESERVED*16777216) + (0*65536) + (0*256) + 3,
     (PC_RESERVED*16777216) + (0*65536) + (0*256) + 4};

hpal = GpiCreatePalette(hab, 0L, LCOLF_CONSECRGB, 4L, aulTable);
```


GpiCreatePS – Create Presentation Space

```
#define INCL_GPICONTROL /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

HPS GpiCreatePS (HAB hab, HDC hdc, PSIZEL pszlSize, ULONG flOptions)

This function creates a presentation space.

Parameters

hab (HAB) – input
Anchor-block handle.

hdc (HDC) – input
Device-context handle.

The handle of a device context with which the presentation space is to be associated, if GPIA_ASSOC is specified. This is mandatory for a micro presentation space (type GPIT_MICRO).

pszlSize (PSIZEL) – input
Presentation-page size.

The size of the presentation page defines a rectangle in presentation page space, with the bottom-left corner at the origin. This rectangle is used for these purposes:

- Together with the page viewport, it defines the device transform. Whenever the presentation space is associated with a device context, a default page viewport is constructed, based on the presentation page size.
- It defines the “area of interest” of the picture. This is recorded in a metafile, if one is generated from this presentation space. Note, however, that depending upon the device transform, information drawn outside it may sometimes be visible; it is *not* a clipping boundary.
- If PU_ARBITRARY is specified, the page viewport is constructed such that the origin of the page rectangle maps to the origin of the default device rectangle (maximized window size, paper size, and so on), and either the right or top edges map, keeping the picture within the default device rectangle, and preserving its aspect ratio.

If 0 is specified as either the width or the height, GPIA_ASSOC must also be specified, and a presentation page of default dimension for the device (see above) is assumed. For PU_ARBITRARY the pel dimensions are used.

flOptions (ULONG) – input
Options.

This contains fields of option bits. For each field, one value should be selected (unless the default is suitable). These values can then be ORed together to generate the parameter.

PS_UNITS

Presentation-page size units.

In each instance, the origin is at the bottom left.

One of these values *must* be specified:

PU_ARBITRARY	Application-convenient units
PU_PELS	Pel coordinates
PU_LOMETRIC	Units of 0.1 mm
PU_HIMETRIC	Units of 0.01 mm
PU_LOENGLISH	Units of 0.01 inch
PU_HIENGLISH	Units of 0.001 inch

GpiCreatePS – Create Presentation Space

PU_TWIPS Units of 1/1440 inch.

PS_FORMAT

Coordinate format.

Indicates options to be used when storing coordinate values internally in the segment store.

For most calls, the format is not directly visible to an application. However, it is visible during editing (for example, GpiQueryElement). The format also has an effect on the amount of storage required for segment store. If a metafile is generated from this presentation space, the format also controls the format of the orders in the metafile.

Note: If GPIF_SHORT is selected, it is the responsibility of the application to ensure that the values passed for graphics coordinates are in the range –32 768 through +32 767, when the drawing mode is **retain** or **draw-and-retain** (see GpiSetDrawingMode), or if a metafile is being created. If in doubt, default or specify GPIF_LONG.

Do not specify GPIF_SHORT if a metafile of unknown format is to be played into this presentation space with GpiPlayMetaFile.

One of these can be selected, for a GPIT_NORMAL presentation space (for a GPIT_MICRO presentation space, only GPIF_DEFAULT is allowed):

GPIF_DEFAULT Default local format (same as GPIF_LONG)

GPIF_SHORT 2-byte integers

GPIF_LONG 4-byte integers.

PS_TYPE

Presentation space.

GPIT_NORMAL Normal presentation space; this is the default

GPIT_MICRO Micro presentation space.

Note: GPIA_ASSOC must also be set if GPIT_MICRO is set.

PS_MODE

Mode. Reserved, must be 0 (default).

PS_ASSOCIATE

Association indicator.

Indicates whether the new presentation space is to be associated with the specified device context:

GPIA_NOASSOC No association is required. This is the default.

GPIA_ASSOC Association with *hdc* required.

Note: GPIA_ASSOC must be set if GPIT_MICRO is set.

Returns

Presentation-space handle:

≠0 Presentation-space handle

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_OR_INCOMPAT_OPTIONS

An invalid or incompatible (with micro presentation space) options parameter was specified with GpiCreatePS or GpiSetPS.

PMERR_DC_IS_ASSOCIATED

An attempt was made to associate a presentation space with a device context that was already associated or to destroy a device context that was associated.

GpiCreatePS —

Create Presentation Space

PMERR_INV_HDC

An invalid device-context handle or (micro presentation space) presentation-space handle was specified.

PMERR_INV_PS_SIZE

An invalid size parameter was specified with GpiCreatePS or GpiSetPS.

Remarks

There are two types of presentation spaces:

- Micro presentation space
- Normal presentation space.

Only a restricted subset of calls is allowed to a micro presentation space; the main difference is that graphic segments (primitives, attributes, and so on) can be retained by the system, for subsequent redraw or editing, in a normal presentation space, but not in a micro presentation space. However, the storage and execution overheads are lower for a micro presentation space.

An initial association of the new presentation space with a device context may be performed (this is mandatory for a micro presentation space), by specifying GPIA_ASSOC.

When a presentation space is associated with a device context, either using this function with GPIA_ASSOC, or explicitly with GpiAssociate, a page viewport in device space is automatically constructed, to which the page is mapped to form the device transform. The value of *PS_UNITS* and the *psiz/Size* parameter, are taken into account.

In general, the size parameter can be safely set to zeroes except when using PU_ARBITRARY units. In that case, use a size in device coordinates obtained from DevQueryCaps. For units other than PU_PELS, a non-zero size can cause a transform to be in effect for the resulting PS.

Related Functions

- GpiAssociate
- GpiDestroyPS
- GpiQueryDevice
- GpiQueryPS
- GpiResetPS
- GpiRestorePS
- GpiSavePS
- GpiSetPageViewport
- GpiSetPS
- WinGetPS
- WinGetScreenPS

GpiCreatePS — Create Presentation Space

Example Code

This example uses the GpiCreatePS function to create a micro presentation space for a memory device context. The function associates the presentation space with the device context and sets the page units to pels. By default, the presentation space is a normal presentation space that uses local storage format.

```
#define INCL_GPICONTROL      /* GPI control Functions      */
#include <os2.h>

HAB hab;                    /* anchor block handle      */
HDC hdc;                    /* device context handle     */
HPS hps;                    /* presentation space handle */
SIZEL sizl = { 0, 0 }; /* use same page size as device */
/*****
 * context data structure *
 *****/
DEVOPENSTRUC dop = {0L, "DISPLAY", NULL, 0L, 0L, 0L, 0L, 0L, 0L};

/* create memory device context */
hdc = DevOpenDC(hab, OD_MEMORY, "", 5L, (PDEVOPENDATA)&dop, NULLHANDLE);

/* Create the presentation and associate the memory device
   context. */
hps = GpiCreatePS(hab, hdc, &sizl, PU_PELS |
                  GPIT_MICRO | GPIA_ASSOC);
```

GpiCreateRegion — Create Region

```
#define INCL_GPIREGIONS /* Or use INCL_GPI or INCL_PM */
```

HRGN GpiCreateRegion (HPS hps, LONG ICount, PRECTL arclRectangles)

This function creates a region, for a particular class of device, using a series of rectangles.

Parameters

hps (HPS) — input

Presentation-space handle.

A region suitable for use with the currently associated device is created.

ICount (LONG) — input

The number of rectangles.

The number specified in *arclRectangles*. If *ICount* is 0, an empty region is created, and *arclRectangles* is ignored.

arclRectangles (PRECTL) — input

An array of rectangles.

The rectangles are specified in device coordinates.

For each rectangle in the array, the value of *xright* must be greater than (or equal to) *xleft*, and *ytop* must be greater than (or equal to) *ybottom*.

Returns

Region handle:

≠0 Region handle

RGN_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

PMERR_INV_COORDINATE

An invalid coordinate value was specified.

PMERR_INV_RECT

An invalid rectangle parameter was specified.

Remarks

The new region is defined by the logical-OR of all of the rectangles specified. Points on the right-hand and top boundaries are not included in the region. Points on the left-hand and bottom boundaries, that are not also on the right-hand or top boundaries (that is, the top-left and bottom-right corner points), are included.

The region is owned by the process from which this function is issued. It cannot be accessed directly from any other process. If it still exists when the process terminates, it is automatically deleted by the system.

GpiCreateRegion – Create Region

Related Functions

- GpiCombineRegion
- GpiDestroyRegion
- GpiEqualRegion
- GpiOffsetRegion
- GpiPaintRegion
- GpiPtInRegion
- GpiQueryRegionBox
- GpiQueryRegionRects
- GpiRectInRegion
- GpiSetRegion

Example Code

This example uses the GpiCreateRegion function to create a region consisting of the union of three rectangles.

```
#define INCL_GPIREGIONS      /* Region functions      */
#include <os2.h>

HPS hps;                    /* presentation space handle */
HRGN hrgn;                  /* handle for region */
RECTL arcl[3] = { 100, 100, 200, 200, /* 1st rectangle */
                  150, 150, 250, 250, /* 2nd rectangle */
                  200, 200, 300, 300 }; /* 3rd rectangle */

hrgn = GpiCreateRegion(hps, /* presentation space */
                      3L,   /* three rectangles */
                      arcl); /* address of array of rectangles */
```

GpiDeleteBitmap – Delete Bit Map

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

BOOL GpiDeleteBitmap (HBITMAP hbm)

This function deletes a bit map.

Parameters

hbm (HBITMAP) – input
Handle of the bit map to be deleted.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HBITMAP

An invalid bit-map handle was specified.

PMERR_BITMAP_IS_SELECTED

An attempt was made to delete a bit map while it was selected into a device context.

PMERR_HBITMAP_BUSY

An internal bit map busy error was detected. The bit map was locked by one thread during an attempt to access it from another thread.

Remarks

There are restrictions on the use of this function while generating a metafile or a PM_Q_STD print file; see "Metafile Restrictions" on page G-1.

Related Functions

- GpiBitBlt
- GpiCreateBitmap
- GpiDrawBits
- GpiLoadBitmap
- GpiQueryBitmapBits
- GpiQueryBitmapDimension
- GpiQueryBitmapHandle
- GpiQueryBitmapParameters
- GpiQueryDeviceBitmapFormats
- GpiSetBitmap
- GpiSetBitmapBits
- GpiSetBitmapDimension
- GpiSetBitmapId
- GpiWCBitBlt
- WinDrawBitmap
- WinGetSysBitmap

GpiDeleteBitmap – Delete Bit Map

Example Code

This example uses the GpiDeleteBitmap function to delete a bit map. The GpiSetBitmap function releases the bit map from the presentation space before deleting it. This is needed only if the bit map is set in the presentation space.

```
#define INCL_GPIBITMAPS          /* GPI Bit map functions */
#include <os2.h>

HPS hps;                        /* presentation space handle */
HBITMAP hbm, hbmPrevious;

hbm = GpiLoadBitmap(hps, 0L, 1, 0L, 0L); /* load the bit map */
hbmPrevious = GpiSetBitmap(hps, hbm);    /* set bit map for PS */

/* bit map displayed with GpiBitBlt */

GpiSetBitmap(hps, hbmPrevious); /* release bit map from PS */
GpiDeleteBitmap(hbm);          /* delete the bit map */
```


GpiDeleteElement — Delete Element

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiDeleteElement (HPS hps)

This function deletes the element indicated by the element pointer.

Parameters

hps (HPS) — input
Presentation-space handle.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.
PMERR_NOT_IN_RETAIN_MODE	An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to retain
PMERR_NO_CURRENT_SEG	An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open segment.
PMERR_INV_IN_ELEMENT	An attempt was made to issue a function invalid inside an element bracket.

Remarks

The element pointer is set to the element immediately preceding the deleted element.

If the element pointer has a value of 0 (points that are logically before the first element), nothing is deleted and the element pointer is not changed.

This function is only valid when the drawing mode (see GpiSetDrawingMode) is set to **retain** (not **draw-and-retain**), and a segment bracket is currently in progress. It is invalid within an element bracket.

GpiDeleteElement – Delete Element

Related Functions

- GpiBeginElement
- GpiDeleteElementRange
- GpiDeleteElementsBetweenLabels
- GpiElement
- GpiEndElement
- GpiLabel
- GpiOffsetElementPointer
- GpiQueryElement
- GpiQueryElementPointer
- GpiQueryElementType
- GpiSetElementPointer
- GpiSetElementPointerAtLabel

Example Code

This example uses the GpiDeleteElement function to delete the third element from the previously created segment 2.

```
#define INCL_GPISEGEDITING      /* GPI Segment Edit functions */
#include <os2.h>

HPS hps;

GpiOpenSegment(hps, 2L);      /* open segment #2      */
GpiSetElementPointer(hps, 3L); /* move to third element */
GpiDeleteElement(hps);        /* delete element      */
GpiCloseSegment(hps);         /* close the segment    */
```

GpiDeleteElementRange — Delete Element Range

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiDeleteElementRange (HPS hps, LONG IFirstElement, LONG ILastElement)

This function deletes all elements between, and including, the elements indicated by the specified element numbers.

Parameters

- hps** (HPS) — input
Presentation-space handle.
- IFirstElement** (LONG) — input
Number of the first element to be deleted.
- ILastElement** (LONG) — input
Number of the last element to be deleted.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.
PMERR_NOT_IN_RETAIN_MODE	An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to retain
PMERR_NO_CURRENT_SEG	An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open segment.
PMERR_INV_IN_ELEMENT	An attempt was made to issue a function invalid inside an element bracket.

Remarks

If either element number is outside the range of the current segment, it is set to the nearest valid value.

When this function has finished, the element pointer is set to the element immediately preceding the deleted element or elements.

This function is only valid when the drawing mode (see GpiSetDrawingMode) is set to **retain** (not **draw-and-retain**), and a segment bracket is currently in progress. It is not valid within an element bracket.

GpiDeleteElementRange — Delete Element Range

Related Functions

- GpiBeginElement
- GpiDeleteElement
- GpiDeleteElementsBetweenLabels
- GpiElement
- GpiEndElement
- GpiLabel
- GpiOffsetElementPointer
- GpiQueryElement
- GpiQueryElementPointer
- GpiQueryElementType
- GpiSetElementPointer
- GpiSetElementPointerAtLabel

Example Code

This example uses the GpiDeleteElementRange function to delete the second through fifth elements in the previously created segment 2.

```
#define INCL_GPISEGEDITING      /* GPI Segment Edit functions */
#include <os2.h>

HPS hps;

GpiOpenSegment(hps, 2L);          /* open segment # 2 */
GpiDeleteElementRange(hps, 2L, 5L); /* delete elements 2 thru 5 */
GpiCloseSegment(hps);            /* close the segment */
```

GpiDeleteElementsBetweenLabels — Delete Elements Between Labels

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiDeleteElementsBetweenLabels (HPS hps, LONG IFirstLabel, LONG ILastLabel)

This function deletes all elements between, but not including, the elements found to contain the indicated labels.

Parameters

hps (HPS) — input

Presentation-space handle.

IFirstLabel (LONG) — input

Label marking the start of the elements to be deleted.

ILastLabel (LONG) — input

Label marking the end of the elements to be deleted.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION

An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_NOT_IN_RETAIN_MODE

An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to **retain**

PMERR_NO_CURRENT_SEG

An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open segment.

PMERR_INV_IN_ELEMENT

An attempt was made to issue a function invalid inside an element bracket.

PMERR_LABEL_NOT_FOUND

The specified element label did not exist.

Remarks

The search for *IFirstLabel* and *ILastLabel* is performed separately, and starts from the element pointed to by the current element pointer.

See also:

- GpiSetElementPointer
- GpiSetElementPointerAtLabel.

If either label cannot be found between the current element pointer location and the end of the segment, an error is generated and no deletion occurs.

GpiDeleteElementsBetweenLabels – Delete Elements Between Labels

On completion, the element pointer is set to the element immediately preceding the deleted elements.

This function is only valid when the drawing mode (see GpiSetDrawingMode) is set to **retain** (not **draw-and-retain**), and a segment bracket is currently in progress. It is not valid within an element bracket.

Related Functions

- GpiBeginElement
- GpiDeleteElement
- GpiDeleteElementRange
- GpiElement
- GpiEndElement
- GpiLabel
- GpiOffsetElementPointer
- GpiQueryElement
- GpiQueryElementPointer
- GpiQueryElementType
- GpiSetElementPointer
- GpiSetElementPointerAtLabel

Example Code

This example uses the GpiDeleteElementsBetweenLabels function to delete the elements between, but not including, the elements having the labels 1 and 2.

```
#define INCL_GPISEGEDITING      /* GPI Segment Edit functions */
#include <os2.h>

HPS hps;

GpiOpenSegment(hps, 2L);        /* open segment #2 */

/* delete elements between 1 and 2 */

GpiDeleteElementsBetweenLabels(hps, 1L, 2L);
GpiCloseSegment(hps);          /* close the segment */
```

GpiDeleteMetaFile —

Delete Metafile

```
#define INCL_GPIMETAFILES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiDeleteMetaFile (HMF hmf)

This function deletes a metafile.

Parameters

hmf (HMF) — input.
Metafile handle.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HMF An invalid metafile handle was specified.

PMERR_METAFILE_IN_USE An attempt has been made to access a metafile that is in use by another thread.

PMERR_TOO_MANY_METAFILES_IN_USE The maximum number of metafiles allowed for a given process was exceeded.

Remarks

This function deletes access to the specified memory metafile and makes the metafile handle invalid.

Related Functions

- GpiCopyMetaFile
- GpiLoadMetaFile
- GpiPlayMetaFile
- GpiQueryMetaFileBits
- GpiQueryMetaFileLength
- GpiSaveMetaFile
- GpiSetMetaFileBits

GpiDeleteMetaFile — Delete Metafile

Example Code

This example uses GpiDeleteMetaFile to delete a metafile previously loaded with GpiLoadMetaFile.

```
#define INCL_GPIMETAFILES      /* Metafile functions      */
#include <os2.h>

BOOL      fSuccess;          /* success indicator      */
HMF      hmf;                /* metafile handle        */
HAB hab;                    /* anchor block handle    */

/* loads metafile from disk */
hmf = GpiLoadMetaFile(hab, "sample.met");
.
.
.

fSuccess = GpiDeleteMetaFile(hmf);
```


GpiDeletePalette — Delete Palette

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiDeletePalette (HPAL hpal)

This function deletes a color palette.

Parameters

hpal (HPAL) — input
Palette handle.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPAL

An invalid color palette handle was specified.

PMERR_PALETTE_SELECTED

Color palette operations cannot be performed on a presentation space while a palette is selected.

PMERR_PALETTE_BUSY

An attempt has been made to reset the owner of a palette when it was busy.

Remarks

The palette must not be currently selected into a presentation space (see GpiSelectPalette).

Related Functions

- GpiAnimatePalette
- GpiCreatePalette
- GpiQueryPalette
- GpiQueryPaletteInfo
- GpiSelectPalette
- GpiSetPaletteEntries
- WinRealizePalette

GpiDeletePalette — Delete Palette

Example Code

This example uses GpiDeletePalette to delete the color palette currently associated with the presentation space, which is determined using GpiQueryPalette.

```
#define INCL_GPILOGCOLORTABLE  /* Color Table functions      */
#include <os2.h>

BOOL    fSuccess;           /* success indicator      */
HPAL    hpal;               /* palette handle         */
HPS     hps;               /* Presentation-space handle */

/* get handle of currently associated palette */
hpal = GpiQueryPalette(hps);

/* delete palette */
fSuccess = GpiDeletePalette(hpal);
```

GpiDeleteSegment — Delete Segment

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiDeleteSegment (HPS hps, LONG ISegId)

This function deletes a retained segment.

Parameters

hps (HPS) — input
Presentation-space handle.

ISegId (LONG) — input
Segment identifier.

The identifier of the segment to be deleted; it must be greater than 0.

Returns

Success indicator:

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME An invalid segment identifier was specified.

PMERR_INV_MICROPS_FUNCTION An attempt was made to issue a function that is invalid in a micro presentation space.

Remarks

If the segment is open when it is deleted, there is no open segment after this function. In this instance, processing as described for GpiCloseSegment is performed.

If the segment is in the segment chain, it is removed from the chain.

This function deletes only a retained segment.

Note: In **draw** drawing mode (see GpiSetDrawingMode), the identifier of the current segment is not remembered, so it is not recognized if specified as the *ISegId* parameter.

GpiDeleteSegment — Delete Segment

Related Functions

- GpiCallSegmentMatrix
- GpiCloseSegment
- GpiCorrelateSegment
- GpiDeleteSegments
- GpiDrawSegment
- GpiErrorSegmentData
- GpiOpenSegment
- GpiQueryInitialSegmentAttrs
- GpiQuerySegmentAttrs
- GpiQuerySegmentNames
- GpiQuerySegmentPriority
- GpiSetInitialSegmentAttrs
- GpiSetSegmentAttrs
- GpiSetSegmentPriority

Example Code

This example uses the GpiDeleteSegment function to delete segment 4, previously created by GpiOpenSegment.

```
#define INCL_GPISEGMENTS      /* Segment functions      */
#include <os2.h>

HPS hps;                      /* presentation space handle */
POINTL ptlStart = { 0, 0 }; /* first vertex */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices */

GpiOpenSegment(hps, 4L);      /* open the segment */
GpiMove(hps, &ptlStart);      /* move to start point (0, 0) */
GpiPolyLine(hps, 3L, ptlTriangle); /* draw triangle */
GpiCloseSegment(hps);         /* close the segment */
.
.
.
GpiDeleteSegment(hps, 4L);     /* delete segment #4 */
```

GpiDeleteSegments —

Delete Segments

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiDeleteSegments (HPS hps, LONG IFirstSegment, LONG ILastSegment)

This function deletes all segments in the given identifier range.

Parameters

hps (HPS) — input

Presentation-space handle.

IFirstSegment (LONG) — input

First identifier in the range; it must be greater than 0.

ILastSegment (LONG) — input

Last identifier in the range; it must be greater than 0.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME

An invalid segment identifier was specified.

PMERR_INV_MICROPS_FUNCTION

An attempt was made to issue a function that is invalid in a micro presentation space.

Remarks

IFirstSegment and *ILastSegment* can have the same value, in which instance, only this segment is deleted. If *IFirstSegment* is greater than *ILastSegment* only the segment with identifier *IFirstSegment* is deleted.

If one of the segments deleted is the currently open segment, there is no open segment after this function. In this instance, processing as described for *GpiCloseSegment* is performed. If any of the segments are in the segment chain, they are removed from the chain.

This function only deletes retained segments.

Note: In **draw** drawing mode (see *GpiSetDrawingMode*), the identifier of the current segment is not remembered, so it is not recognized if it occurs within the range of specified identifiers.

GpiDeleteSegments – Delete Segments

Related Functions

- GpiCallSegmentMatrix
- GpiCloseSegment
- GpiCorrelateSegment
- GpiDeleteSegment
- GpiDrawSegment
- GpiErrorSegmentData
- GpiOpenSegment
- GpiQueryInitialSegmentAttrs
- GpiQuerySegmentAttrs
- GpiQuerySegmentNames
- GpiQuerySegmentPriority
- GpiSetInitialSegmentAttrs
- GpiSetSegmentAttrs
- GpiSetSegmentPriority

Example Code

This example uses the GpiDeleteSegments function to delete segments 4 through 6, created by GpiOpenSegment.

```
#define INCL_GPISEGMENTS      /* Segment functions      */
#include <os2.h>

HPS hps;                      /* presentation space handle */

GpiOpenSegment(hps, 4L);      /* open segment 4            */
GpiCloseSegment(hps);         /* close the segment         */
GpiOpenSegment(hps, 5L);      /* open segment 5            */
GpiCloseSegment(hps);         /* close the segment         */
GpiOpenSegment(hps, 6L);      /* open segment 6            */
GpiCloseSegment(hps);         /* close the segment         */
.
.
.
GpiDeleteSegments(hps, 4L, 6L); /* delete segments 4 through 6 */
```

GpiDeleteSetId – Delete Set Identifier

```
#define INCL_GPILCIDS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiDeleteSetId (HPS hps, LONG lLcid)

This function deletes a logical font or bit-map tag.

Parameters

hps (HPS) – input
Presentation-space handle.

lLcid (LONG) – input
Local identifier.

The local identifier (lcid) for the object.

If LCID_ALL is specified, all logical fonts are deleted, and all bit-map tagging is removed. If LCID_DEFAULT or LCID_ALL is specified, the original default font is restored if it has been changed (see GpiCreateLogFont).

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_SETID	An invalid setid parameter was specified.
PMERR_SETID_NOT_FOUND	An attempt was made to delete a setid that did not exist.
PMERR_SETID_IN_USE	An attempt was made to specify a setid that was already in use as the currently selected character, marker or pattern set.

Remarks

If the object is a logical font, it is deleted, and is no longer available for use. If the object is a bit map, it is no longer tagged with the local identifier; the bit map is not deleted and its handle remains valid.

In either instance, the *lLcid* is released and is now available for reuse, unless the object is currently selected (as the current character, pattern, or marker set), in which instance an error is raised.

If this function occurs within a path definition when the drawing mode (see GpiSetDrawingMode) is **retain** or **draw-and-retain**, its effect is not stored with the definition.

Note: This function must not be used when creating SAA-conforming metafiles; see “Metafile Restrictions” on page G-1.

Related Functions

- GpiCreateLogFont
- GpiLoadFonts
- GpiQueryFontMetrics
- GpiQueryFonts
- GpiQueryKerningPairs
- GpiQueryNumberSetIds
- GpiQuerySetIds
- GpiQueryWidthTable
- GpiUnloadFonts
- GpiSetBitmapId
- GpiSetCharSet

Example Code

This example uses the GpiDeleteSetId function to delete a logical font. The GpiSetCharSet function is required only if the logical font is the current font for the presentation space.

```
#define INCL_GPILCIDS          /* Font functions          */
#define INCL_GPIPRIMITIVES    /* GPI primitive functions */
#include <os2.h>

HPS hps;          /* presentation space handle */
FATTRS fat;

/* create and set the font */

GpiCreateLogFont(hps, NULL, 1L, &fat);
GpiSetCharSet(hps, 1L);
.
.
.
GpiSetCharSet(hps, 0L);          /* release the font before deleting */
GpiDeleteSetId(hps, 1L);        /* delete the logical font          */
```


GpiDestroyPS – Destroy Presentation Space

```
#define INCL_GPICONTROL /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

```
BOOL GpiDestroyPS (HPS hps)
```

This function destroys the presentation space.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_PS_IS_ASSOCIATED

An attempt was made to destroy a presentation or associate a presentation space that is still associated with a device context.

Remarks

All resources owned by the presentation space are released, and any subsequent calls that use the value of the presentation space handle are rejected.

Related Functions

- GpiAssociate
- GpiCreatePS
- GpiQueryDevice
- GpiQueryPS
- GpiResetPS
- GpiRestorePS
- GpiSavePS
- GpiSetPS

GpiDestroyPS – Destroy Presentation Space

Example Code

This example uses the GpiDestroyPS function to destroy the presentation space associated with a memory device context.

```
#define INCL_GPICONTROL      /* GPI control Functions      */
#define INCL_DEV             /* Device Function definitions */
#include <os2.h>

HAB      hab;      /* Anchor-block handle      */
HPS      hps;      /* Target presentation-space handle */
HDC      hdc;      /* Device-context handle     */
DEVOPENSTRUC dop;  /* context data structure    */
SIZEL page = { 0, 0 }; /* page size (use same as device) */

/* Create the memory device context and presentation space. */
hdc = DevOpenDC(hab, OD_MEMORY, "", 5L, (PDEVOPENDATA)&dop, NULLHANDLE);
hps = GpiCreatePS(hab, hdc, &page, PU_PELS|GPIT_MICRO|GPIA_ASSOC);
.
.
.
GpiAssociate(hps, NULLHANDLE); /* disassociate device context */
GpiDestroyPS(hps);           /* destroys presentation space */
DevCloseDC(hdc);             /* closes device context      */
```

GpiDestroyRegion – Destroy Region

```
#define INCL_GPIREGIONS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiDestroyRegion (HPS hps, HRGN hrgn)

This function destroys a region.

Parameters

hps (HPS) – input
Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

hrgn (HRGN) – input
Handle of region to be destroyed.

If this is NULLHANDLE, the call takes no action, and completes without error.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_HRGN	An invalid region handle was specified.
PMERR_REGION_IS_CLIP_REGION	An attempt was made to perform a region operation on a region that is selected as a clip region.
PMERR_HRGN_BUSY	An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

Remarks

This function cannot be used to destroy the clip region; the clip region must first be deselected with GpiSetClipRegion.

GpiDestroyRegion – Destroy Region

Related Functions

Prerequisite Functions

- GpiSetClipRegion(if the region to be destroyed is a clip region)

Other Related Functions

- GpiCombineRegion
- GpiCreateRegion
- GpiEqualRegion
- GpiOffsetRegion
- GpiPaintRegion
- GpiPtInRegion
- GpiQueryRegionBox
- GpiQueryRegionRects
- GpiRectInRegion
- GpiSetRegion

Example Code

This example uses the GpiDestroyRegion function to destroy a region after drawing a complex figure.

```
#define INCL_GPIREGIONS          /* Region functions          */
#include <os2.h>

HPS hps;                        /* presentation space handle */
HRGN hrgn;
RECTL arc1[3] = { 10,10,20,20,15,15,25,25,20,20,30,30 };

hrgn = GpiCreateRegion(hps, 3L, arc1); /* use 3 rectangles */
GpiPaintRegion(hps, hrgn);           /* paint the region */
GpiDestroyRegion(hps, hrgn);         /* destroy the region */
```

GpiDrawBits – Draw Bits

#define INCL_GPIBITMAPS /* Or use INCL_GPI or INCL_PM. Also in COMMON section */

**LONG GpiDrawBits (HPS hpsTarget, PVOID pBits, PBITMAPINFO2 pbmi2InfoTable,
LONG ICount, PPOINTL aptlPoints, LONG IRop, ULONG fOptions)**

This function draws a rectangle of bits.

Parameters

hpsTarget (HPS) – input
Target presentation-space handle.

pBits (PVOID) – input
Source bits.

The source bits must be in one of the standard bit-map formats.

pbmi2InfoTable (PBITMAPINFO2) – input
Bit-map information table.

This describes the format of the source bits.

ICount (LONG) – input
Point count.

This count must be equal to 4.

aptlPoints (PPOINTL) – input
Point array

Array of *ICount* points, in the order **Tx1, Ty1, Tx2, Ty2, Sx1, Sy1, Sx2, Sy2**. These are:

Tx1,Ty1 Specify the bottom left corner of the target rectangle in target world coordinates.

Tx2,Ty2 Specify the top right corner of the target rectangle in target world coordinates.

An error occurs if the left coordinate of the target rectangle is greater than the right, or if the bottom coordinate is greater than the top.

Sx1,Sy1 Specify the bottom left corner of the source rectangle in source device coordinates.

Sx2,Sy2 Specify the top right corner of the source rectangle in source device coordinates.

IRop (LONG) – input
Mixing function required.

Each plane of the target can be considered to be processed separately. For any pel in a target plane, three bits together with the *IRop* values are used to determine the final value. These are the value of that pel in the Pattern (P) and Source (S) data and the initial value of that pel in the Target (T) data. For any combination of P, S, and T pel values, the final target value for the pel is determined by the appropriate *IRop* bit value as shown below:

P	S	T(initial)	T(final)
0	0	0	Index bit 0 (least significant)
0	0	1	Index bit 1
0	1	0	Index bit 2
0	1	1	Index bit 3
1	0	0	Index bit 4
1	0	1	Index bit 5
1	1	0	Index bit 6
1	1	1	Index bit 7 (most significant)

GpiDrawBits – Draw Bits

The index formed as described above determines the mixing required. Mnemonic names are available for commonly used mixes:

ROP_SRCOPY	/* SRC	*/
ROP_SRCPAINT	/* SRC OR DST	*/
ROP_SRCAND	/* SRC AND DST	*/
ROP_SRCINVERT	/* SRC XOR DST	*/
ROP_SRCERASE	/* SRC AND NOT(DST)	*/
ROP_NOTSRCCOPY	/* NOT(SRC)	*/
ROP_NOTSRCERASE	/* NOT(SRC) AND NOT(DST)	*/
ROP_MERGECOPY	/* SRC AND PAT	*/
ROP_MERGEPAINT	/* NOT(SRC) OR DST	*/
ROP_PATCOPY	/* PAT	*/
ROP_PATPAINT	/* NOT(SRC) OR PAT OR DST	*/
ROP_PATINVERT	/* DST XOR PAT	*/
ROP_DSTINVERT	/* NOT(DST)	*/
ROP_ZERO	/* 0	*/
ROP_ONE	/* 1	*/

fOptions (ULONG) – input
Options.

How eliminated lines or columns are treated if a compression is performed.

Flags 15 through 31 of *fOptions* can be used for privately supported modes for particular devices.

BBO_OR The default value; if compression is necessary, logical-OR eliminated rows or columns. This is useful for white on black.

BBO_AND If compression is necessary, logical-AND eliminated rows or columns. This is useful for black on white.

BBO_IGNORE If compression is necessary, ignore eliminated rows or columns. This is useful for color.

Returns

Correlation and error indicators:

GPI_OK Successful

GPI_HITS Correlate hits

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT An invalid length or count parameter was specified.

PMERR_INV_BITBLT_MIX An invalid *IRop* parameter was specified with a GpiBitBlt or GpiWCBitBlt function.

PMERR_INV_BITBLT_STYLE An invalid options parameter was specified with a GpiBitBlt or GpiWCBitBlt function.

PMERR_INV_COORDINATE An invalid coordinate value was specified.

PMERR_INV_RECT An invalid rectangle parameter was specified.

PMERR_INCORRECT_DC_TYPE An attempt was made to perform a bit-map operation on a presentation space associated with a device context of a type that is unable to support bit-map operations.

GpiDrawBits —

Draw Bits

Remarks

A rectangle of bit-map image data is copied from storage to a bit map selected into a device context associated with the target presentation space. Alternatively, the target presentation space can be associated with a device context that specifies a suitable raster device, for example, the screen. An error occurs if this device does not support raster operations.

The source bits must be in one of the standard bit-map formats.

A rectangle is specified in device coordinates for the source bits, and one in world coordinates for the target presentation space. The source rectangle is noninclusive; the left and lower boundaries in device space are included, but not the right and upper boundaries. Thus if the bottom left is equal to the top right, the rectangle is deemed to be empty. The target rectangle is “inclusive-inclusive”; that is, all boundaries are included in the rectangle.

If the target rectangle, after transformation to device coordinates and adjustment for inclusivity, is not the same size as the source rectangle, then stretching or compressing of the data occurs. *flOptions* specifies how eliminated rows or columns of bits are to be treated if compression occurs. Note that the pattern data is never stretched or compressed.

These current attributes of the target presentation space are used (other than for converting between monochrome and color, as described below):

- Area color
- Area background color
- Pattern set
- Pattern symbol.

The color values are used in conversion between monochrome and color data. This is the only format conversion performed by this function. The conversions are:

- Output of a monochrome pattern to a color device

In this instance the pattern is converted first to a color pattern, using the current area colors:

- source 1s → area foreground color
- source 0s → area background color.

- Copying from a monochrome bit map to a color bit map (or device)

The source bits are converted as follows:

- source 1s → image foreground color
- source 0s → image background color.

- Copying from a color bit map to a monochrome bit map (or device)

The source bits are converted as follows:

- source nonzeros → image foreground color
- source 0s → image background color.

If the mix (*IRop*) does not call for a pattern, the pattern set and pattern symbol are not used.

Neither the source nor the pattern is required when a bit map, or part of a bit map, is to be cleared to a particular color.

If the mix does require both source and pattern, a three-way operation is performed.

If a pattern is required, dithering may be performed for solid patterns in a color that is not available on the device. See *GpiSetPattern*.

GpiDrawBits – Draw Bits

This function can cause immediate drawing, or be retained in segment store, or both of these, depending upon the drawing mode (see GpiSetDrawingMode). If the function is retained in segment store, the storage identified by the *pBits* and *pbmi2InfoTable* parameters must not be changed or freed by the application while the segment containing the function can still be drawn. However, if a metafile is generated and no further drawing is needed, this does not apply, as the information is encapsured in the metafile.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see “Metafile Restrictions” on page G-1.

Related Functions

- GpiBitBlt
- GpiCreateBitmap
- GpiDeleteBitmap
- GpiLoadBitmap
- GpiQueryBitmapBits
- GpiQueryBitmapDimension
- GpiQueryBitmapHandle
- GpiQueryBitmapParameters
- GpiQueryDeviceBitmapFormats
- GpiSetBitmap
- GpiSetBitmapBits
- GpiSetBitmapDimension
- GpiSetBitmapId
- GpiWCBitBlt
- WinDrawBitmap
- WinGetSysBitmap

Graphic Elements and Orders

Element Type: OCODE_GBBLT

Order: Bitblt

GpiDrawBits — Draw Bits

Example Code

This example uses GpiDrawBits to draw a rectangle of bits. The bit map was previously placed in application memory using GpiQueryBitmapBits; when the stored image is displayed, it will be a compressed copy (ROP_SRCCOPY) of the source bit map (note the difference between the target and source rectangle sizes), with eliminated rows/columns ignored (BBO_IGNORE) when compression takes place.

```
#define INCL_GPIBITMAPS          /* Bit map functions          */
#include <os2.h>

HPS    hps;          /* presentation-space handle */
PBYTE  pb;           /* bit-map image data        */
BITMAPINFO2 pbmi;     /* bit-map information table  */
LONG    lHits;        /* correlation/error indicator */
LONG    lScan;        /* number of lines scanned    */
/* target and source rectangles */
POINTL  aptlPoints[4]={ 300, 400, 350, 450, 0, 0, 100, 100 };

/* scan and transfer bit map to application storage */
pbmi.cbFix = 16L;
pbmi.cPlanes = 1;
pbmi.cbBitCount = 4;
lScan = GpiQueryBitmapBits(hps, 0L, 100L, pb, &pbmi);
.
.
.
/* draw stored rectangle bit map */
lHits = GpiDrawBits(hps, (VOID *)pb, &pbmi, 4L,
                    aptlPoints, ROP_SRCCOPY, BBO_IGNORE);
```

GpiDrawChain – Draw Chain

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiDrawChain (HPS hps)

This function draws the segments that are in the segment chain.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Success indicator:
TRUE Successful completion
FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.

Remarks

The segments drawn are all of the retained segments that have the ATTR_CHAINED segment attribute (see GpiSetInitialSegmentAttrs), together with all of the unchained segments that are called from them.

The drawing operation is controlled by the calls set by the draw controls (see GpiSetDrawControl), except for the correlate control. If there is not a segment open at the time of the draw, and this function is followed by primitives or attributes, without first opening a segment, the processing is as described for GpiCloseSegment.

If a segment is already open at the time of the draw, GpiCloseSegment processing is not performed at the completion of the draw (except that any unclosed path or area is abandoned with an error). In this instance, if the open segment is the last one drawn (and no dynamic segments had to be drawn), attributes and other parameters are in the correct state to continue drawing in any drawing mode.

Dynamic segments are not drawn if they are found while processing the segment chain. However, depending on the setting of DCTL_DYNAMIC (see GpiSetDrawControl), they may be removed before, and drawn after, the operation.

It may be necessary to ensure that attributes, model transform, current position, and viewing limits are reset to their default values, before processing the chain. This can be done by ensuring that the first segment to be drawn does not have the ATTR_FASTCHAIN attribute (see GpiSetInitialSegmentAttrs), or by issuing GpiResetPS before the GpiDrawChain. The latter method also resets the clip path to no clipping, which may also be necessary.

GpiDrawChain —

Draw Chain

It is an error to issue this function while any of these brackets are open:

- Area bracket
- Path bracket
- Element bracket.

Related Functions

- GpiDrawDynamics
- GpiDrawFrom
- GpiDrawSegment
- GpiErase
- GpiQueryDrawControl
- GpiQueryDrawingMode
- GpiQueryStopDraw
- GpiRemoveDynamics
- GpiSetDrawControl
- GpiSetDrawingMode
- GpiSetStopDraw

Example Code

This function uses GpiDrawChain to draw the two chained segments.

```
#define INCL_GPISEGMENTS      /* Segment functions      */
#include <os2.h>

BOOL    fSuccess;           /* success indicator      */
HPS     hps;                /* presentation-space handle */

/* The chaining attribute is switched on */
GpiSetInitialSegmentAttrs(hps, ATTR_CHAINED, ATTR_ON);

/* two chained segments are defined */
GpiOpenSegment(hps, 1L);
.
.
GpiCloseSegment(hps);

GpiOpenSegment(hps, 2L);
.
.
GpiCloseSegment(hps);

/* draw the segment chain */
fSuccess = GpiDrawChain(hps);
```

GpiDrawDynamics – Draw Dynamics

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiDrawDynamics (HPS hps)

This function redraws the dynamic segments in, or called from, the segment chain.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION

An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_FOR_THIS_DC_TYPE

An attempt has been made to issue GpiRemoveDynamics or GpiDrawDynamics to a presentation space associated with a metafile device context.

Remarks

Dynamic segments are those segments in the segment chain that have the ATTR_DYNAMIC segment attribute (see GpiSetInitialSegmentAttrs). It is preferable to position dynamic segments at the start of the segment chain.

Dynamic segments can either be drawn with this function, or by setting the DCTL_DYNAMIC draw control (see GpiSetDrawControl), and issuing one of the other GpiDraw... calls.

If there is no range set by a previous GpiRemoveDynamics, all dynamic segments are redrawn by GpiDrawDynamics). However, if GpiRemoveDynamics specified a range in the segment chain, the redraw is restricted to the dynamic segments that are in, or called from, the selected range. (See GpiRemoveDynamics).

Note: The redraw is controlled by the calls set by previous calls to GpiSetDrawControl.

The “stop draw” condition can be set (from another thread) while GpiDrawDynamics is in progress. This is useful in responding to a new position by setting this condition, and then clearing it and redrawing at the new position.

If “Erase before draw” is set ON (see GpiSetDrawControl), the presentation space is erased before the redraw.

It may be necessary to ensure that attributes, model transform, current position, and viewing limits are reset to their default values, before processing the segments. This can be done either by ensuring that the first dynamic segment to be drawn does not have the ATTR_FASTCHAIN attribute

GpiDrawDynamics —

Draw Dynamics

(see `GpiSetInitialSegmentAttrs`), or by issuing `GpiResetPS` before the `GpiDrawDynamics`. The latter method also resets the clip path to no clipping, which may also be necessary.

If this function is followed by primitives or attributes, without first opening a segment, the processing is as described for `GpiCloseSegment`. In particular, note that during `GpiDrawDynamics`, the system forces the foreground mix to `FM_XOR` and the background mix to `BM_LEAVEALONE`. It may be necessary to set one or both of these before starting to draw.

Related Functions

- `GpiDrawChain`
- `GpiDrawFrom`
- `GpiDrawSegment`
- `GpiErase`
- `GpiGetData`
- `GpiPutData`
- `GpiQueryDrawControl`
- `GpiQueryDrawingMode`
- `GpiQueryStopDraw`
- `GpiRemoveDynamics`
- `GpiSetDrawControl`
- `GpiSetDrawingMode`
- `GpiSetInitialSegmentAttrs`
- `GpiSetSegmentAttrs`
- `GpiSetStopDraw`

Example Code

This example uses `GpiDrawDynamics` to redraw the two previously defined dynamic chained segments.

```
#define INCL_GPISEGMENTS      /* Segment functions      */
#include <os2.h>

BOOL      fSuccess;          /* success indicator      */
HPS       hps;               /* presentation-space handle */

/* The chaining attribute is switched on */
GpiSetInitialSegmentAttrs(hps, ATTR_CHAINED | ATTR_DYNAMIC,
                          ATTR_ON);

/* two dynamic chained segments are defined */
GpiOpenSegment(hps, 1L);
.
.
GpiCloseSegment(hps);

GpiOpenSegment(hps, 2L);
.
.
GpiCloseSegment(hps);

/* draw the dynamic segment chain */
fSuccess = GpiDrawDynamics(hps);
```

GpiDrawFrom – Draw From

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiDrawFrom (HPS hps, LONG IFirstSegment, LONG ILastSegment)

This function draws a section of the segment chain.

Parameters

hps (HPS) – input
Presentation-space handle.

IFirstSegment (LONG) – input
First segment to be drawn; it must be greater than 0.

ILastSegment (LONG) – input
Last segment to be drawn; it must be greater than 0.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.
PMERR_SEG_NOT_FOUND	The specified segment identifier did not exist
PMERR_SEG_NOT_CHAINED	An attempt was made to issue GpiDrawFrom, GpiCorrelateFrom or GpiQuerySegmentPriority for a segment that was not chained.
PMERR_INV_SEG_NAME	An invalid segment identifier was specified.

Remarks

Drawing starts at the segment identified by *IFirstSegment* and includes all chained segments (those with the ATTR_CHAINED segment attribute, see GpiSetInitialSegmentAttrs), and the segments called from them, up to, and including, the segment identified by *ILastSegment*.

The drawing operation is controlled by the calls set by the draw controls (see GpiSetDrawControl), except for the correlate control.

If there is not a segment open at the time of the draw, and this function is followed by primitives or attributes, without first opening a segment, the processing is as described for GpiCloseSegment.

If a segment is already open at the time of the draw, GpiCloseSegment processing is not performed at the completion of the draw (except that any unclosed path or area is terminated with an error). In this instance, if the open segment is the last one drawn (and no dynamic segments had to be drawn), attributes and other parameters are in the correct state to continue drawing in any drawing mode.

GpiDrawFrom — Draw From

Dynamic segments are not drawn if they are found while processing the segment chain. However, depending on the setting of DCTL_DYNAMIC (see GpiSetDrawControl), they may be removed before, and drawn after, the operation. If this happens, then *all* dynamic segments are involved, whether they occur within the range specified or not.

It may be necessary to ensure that attributes, model transform, current position, and viewing limits are reset to their default values, before processing the segments. This can be done either by ensuring that the first segment to be drawn does not have the ATTR_FASTCHAIN attribute (see GpiSetInitialSegmentAttrs), or by issuing GpiResetPS before the GpiDrawFrom. The latter method also resets the clip path to no clipping, which may also be necessary.

It is an error to issue this function while any of these brackets are open:

- Area bracket
- Path bracket
- Element bracket.

If *IFirstSegment* does not exist, or is not in the segment chain, an error is raised. If the *ILastSegment* does not exist, or is not in the chain, or is chained before the *IFirstSegment*, no error is raised, and processing continues to the end of the chain.

Related Functions

- GpiDrawChain
- GpiDrawDynamics
- GpiDrawSegment
- GpiErase
- GpiGetData
- GpiPutData
- GpiQueryDrawControl
- GpiQueryDrawingMode
- GpiQueryStopDraw
- GpiRemoveDynamics
- GpiSetDrawControl
- GpiSetDrawingMode
- GpiSetStopDraw

Example Code

This example uses the GpiDrawFrom function to draw all segments in the picture chain between and including the segments 1 and 4.

```
#define INCL_GPISEGMENTS      /* Segment functions      */
#include <os2.h>

HPS hps;                      /* presentation space handle */

GpiDrawFrom(hps, 1L, 4L);
```

GpiDrawSegment — Draw Segment

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiDrawSegment (HPS hps, LONG ISegment).

This function draws the specified segment.

Parameters

hps (HPS) — input
Presentation-space handle.

ISegment (LONG) — input
Segment to be drawn; it must be greater than 0.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_SEG_NOT_FOUND The specified segment identifier did not exist

PMERR_INV_SEG_NAME An invalid segment identifier was specified.

Remarks

The drawing operation is controlled by the calls set by the draw controls (see GpiSetDrawControl), except for the correlate control.

If there is not a segment open at the time of the draw, and this function is followed by primitives or attributes, without first opening a segment, the processing is as described for GpiCloseSegment.

If a segment is already open at the time of the draw, GpiCloseSegment processing is not performed at the completion of the draw (except that any unclosed path or area is abandoned with an error). In this instance, if the open segment is the segment specified in *ISegment*, and no dynamic segments had to be drawn, then attributes and other parameters are in the correct state to continue drawing in any drawing mode.

Depending on the setting of DCTL_DYNAMIC (see GpiSetDrawControl), all of the dynamic segments in the chain may be removed before, and drawn after, the specified segment is drawn. (Note that if the specified segment is itself dynamic, it is only drawn in this way.)

This function differs from the other GpiDraw... calls, in that the segment to be drawn need not be a chained segment.

It may be necessary to ensure that attributes, model transform, current position, and viewing limits are reset to their default values, before processing the segment. This can be done either by ensuring that the segment does not have the ATTR_FASTCHAIN attribute (see

GpiDrawSegment —

Draw Segment

GpiSetInitialSegmentAttrs), or by issuing GpiResetPS before the GpiDrawSegment. The latter method also resets the clip path to no clipping, which may also be necessary.

It is an error to issue this function while any of these brackets are open:

- Area bracket
- Path bracket
- Element bracket.

Related Functions

- GpiDrawChain
- GpiDrawDynamics
- GpiDrawFrom
- GpiErase
- GpiErrorSegmentData
- GpiQueryDrawControl
- GpiQueryDrawingMode
- GpiQueryStopDraw
- GpiRemoveDynamics
- GpiSetDrawControl
- GpiSetDrawingMode
- GpiSetStopDraw

Example Code

This example uses the GpiDrawSegment function to draw segment 4.

```
#define INCL GPISEGMENTS      /* Segment functions      */
#include <os2.h>

HPS hps;                    /* presentation space handle */
POINTL ptlStart = { 0, 0 }; /* first vertex              */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices */

GpiOpenSegment(hps, 4L);    /* open the segment          */
GpiMove(hps, &ptlStart);    /* move to start point (0, 0) */
GpiPolyLine(hps, 3L, ptlTriangle); /* draw triangle          */
GpiCloseSegment(hps);      /* close the segment          */
.
.
.
GpiDrawSegment(hps, 4L);    /* draw segment #4          */
```

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI or INCL_PM */
```

```
LONG GpiElement (HPS hps, LONG IType, PSZ pszDesc, LONG ILength, PBYTE pbData)
```

This function adds a single element to the current segment.

Parameters

hps (HPS) – input

Presentation-space handle.

IType (LONG) – input

Type to be associated with the element.

Application-defined elements should have type values in the range X'81xxxxxx' through X'FFxxxxxx' so as to avoid conflict with system-generated elements.

pszDesc (PSZ) – input

Element description.

This is a variable length character string that is recorded with the element.

ILength (LONG) – input

Length of content data for the element.

This must not be greater than 63KB.

pbData (PBYTE) – input

Buffer pointer.

Element content data.

Returns

Correlation and error indicators:

GPI_OK Successful

GPI_HITS Correlate hits

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION

An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

PMERR_DATA_TOO_LONG

An attempt was made to transfer more than the maximum permitted amount of data (64512 bytes) using GpiPutData, GpiGetData, or GpiElement.

PMERR_ALREADY_IN_ELEMENT

An attempt was made to begin a new element while an existing element bracket was already open.

GpiElement — Element

Remarks

The element is stored in the current segment if the drawing mode (see GpiSetDrawingMode) is **retain** or **draw-and-retain**. It is drawn if the drawing mode is **draw** or **draw-and-retain**.

It is an error if the element data contains any begin or end element orders. Similarly, this function is not valid within an element bracket.

Note: No coordinate conversion is performed by this function. The application must ensure that the coordinates within the element are in the correct format for the presentation space (see GpiCreatePS).

Related Functions

- GpiBeginElement
- GpiDeleteElement
- GpiDeleteElementRange
- GpiDeleteElementsBetweenLabels
- GpiEndElement
- GpiLabel
- GpiOffsetElementPointer
- GpiQueryElement
- GpiQueryElementPointer
- GpiQueryElementType
- GpiSetElementPointer
- GpiSetElementPointerAtLabel

Example Code

This example uses GpiElement to add a single element to the current segment: an arc starting at the current position, passing through (10,10), and ending at (5,5).

```
#define INCL_GPISEGEDITING      /* GPI Segment Edit functions */
#define INCL_GPISEGMENTS      /* Segment functions */
#define INCL_ORDERS            /* Graphical Order Formats */
#include <os2.h>

LONG      lHits;              /* correlation/error indicator */
HPS       hps;               /* presentation-space handle */
LONG      lType;             /* element type */
char      pszDesc[4];        /* element description */
LONG      lLength;           /* length of element data */
LORDER    pbData;            /* pointer to element data */
ORDERL_GCARC lArcPts = {10L,10L,5L,5L}; /* arc points structure */

GpiOpenSegment(hps, 3L); /* opens segment to receive element */

/* type is order code for arc at current position (GARC) */
lType = OCODE_GCARC;

/* call the element 'Arc' */
strcpy(pszDesc,"Arc");

/* length of element data */
lLength = sizeof(LORDER);

/* fill element data structure */
pbData.idCode = OCODE_GCARC; /* order code: arc at current
                             position */
pbData.uchLength = sizeof(ORDERL_GCARC);
/* order data contains arc points structure */
memcpy(pbData.uchData, lArcPts, sizeof(ORDERL_GCARC));

/* add element */
lHits = GpiElement(hps, lType, pszDesc, lLength, (BYTE *)&pbData);

GpiCloseSegment(hps); /* closes segment that received data */
```

GpiEndArea – End Area

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

LONG GpiEndArea (HPS hps)

This function ends the construction of a shaded area.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Correlation and error indicators:

GPI_OK	Successful
GPI_HITS	Correlate hits
GPI_ERROR	Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_NOT_IN_AREA	An attempt was made to end an area using GpiEndArea or during segment drawing while not in an area bracket.
PMERR_COORDINATE_OVERFLOW	An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

Remarks

The construction is started by the GpiBeginArea function. If necessary, a final line is constructed (to the starting point of the last figure) to close the area.

The current position is not changed, unless a closure line has to be drawn, in which case the current position is moved to the end point of the line.

Related Functions

Prerequisite Functions

- GpiBeginArea

Other Related Functions

- GpiSetPattern
- GpiSetPatternRefPoint
- GpiSetPatternSet
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

Graphic Elements and Orders

Element Type: OCODE_GEAR

Order: End Area

Example Code

This example uses the GpiEndArea function to end an area bracket. The function draws the area (a triangle) by filling the outline with the current fill pattern.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions */
#include <os2.h>

HPS hps;                        /* presentation space handle */
POINTL ptlStart = { 0, 0 }; /* first vertex */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices */

GpiBeginArea(hps, BA_NOBOUNDARY | BA_ALTERNATE);
GpiMove(hps, &ptlStart);
GpiPolyLine(hps, 3L, ptlTriangle);
GpiEndArea(hps);
```

GpiEndElement — End Element

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiEndElement (HPS hps)

This function terminates an element started by GpiBeginElement.

Parameters

hps (HPS) — input
Presentation-space handle.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_NOT_IN_ELEMENT

An attempt was made to end an element using GpiEndElement or during segment drawing while not in an element bracket.

Related Functions

Prerequisite Functions

- GpiBeginElement

Other Related Functions

- GpiDeleteElement
- GpiDeleteElementRange
- GpiDeleteElementsBetweenLabels
- GpiElement
- GpiLabel
- GpiOffsetElementPointer
- GpiQueryElement
- GpiQueryElementPointer
- GpiQueryElementType
- GpiSetElementPointer
- GpiSetElementPointerAtLabel

GpiEndElement – End Element

Example Code

This example uses the GpiEndElement function to end an element bracket.

```
#define INCL_GPISEGEDITING      /* GPI Segment Edit functions */
#include <os2.h>

HPS hps;                       /* presentation space handle */
POINTL ptlStart = { 0, 0 }; /* first vertex */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices */

.
.
/* begin the element bracket */
GpiBeginElement(hps, 1L, "Triangle");
GpiMove(hps, &ptlStart);          /* move to start point (0, 0) */
GpiPolyLine(hps, 3L, ptlTriangle); /* draw triangle */
GpiEndElement(hps);               /* end element bracket */
```


GpiEndPath — End Path

```
#define INCL_GPIPATHS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiEndPath (HPS hps)

This function ends the specification of a path started by GpiBeginPath.

Parameters

hps (HPS) — input
Presentation-space handle.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_NOT_IN_PATH

An attempt was made to end a path using GpiEndPath or during segment drawing while not in a path bracket.

Related Functions

Prerequisite Functions

- GpiBeginPath

Other Related Functions

- GpiFillPath
- GpiModifyPath
- GpiOutlinePath
- GpiPathToRegion
- GpiSetClipPath
- GpiStrokePath
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

Graphic Elements and Orders

Element Type: OCODE_GEPTH

Order: End Path

Example Code

This example uses the GpiEndPath function to end a path bracket. When the path bracket is ended, a subsequent call to the GpiFillPath function draws and fills the path.

```
#define INCL_GPIPATHS          /* GPI Path functions          */
#include <os2.h>

HPS hps;                      /* presentation space handle */
POINTL ptlStart = { 0, 0 }; /* first vertex                */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices */

GpiBeginPath(hps, 1L);        /* start the path bracket */
GpiMove(hps, &ptlStart);      /* move to starting point */
GpiPolyLine(hps, 2L, ptlTriangle); /* draw the three sides */
GpiCloseFigure(hps);          /* close the triangle */
GpiEndPath(hps);              /* end the path bracket */
GpiFillPath(hps, 1L, FPATH_ALTERNATE); /* draw and fill the path */
```

GpiEqualRegion — Equal Region

```
#define INCL_GPIREGIONS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiEqualRegion (HPS hps, HRGN hrgnSrc1, HRGN hrgnSrc2)

This function checks whether two regions are identical.

Parameters

hps (HPS) — input

Presentation-space handle.

The regions must be owned by the device identified by the currently associated device context.

hrgnSrc1 (HRGN) — input

Handle of first region.

hrgnSrc2 (HRGN) — input

Handle of second region.

Returns

Equality and error indicators:

EQRGN_NOTEQUAL Not equal

EQRGN_EQUAL Equal

EQRGN_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HRGN

An invalid region handle was specified.

PMERR_REGION_IS_CLIP_REGION

An attempt was made to perform a region operation on a region that is selected as a clip region.

PMERR_HRGN_BUSY

An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

Remarks

Both regions must be of the same device class. It is invalid if the specified region is currently selected as the clip region (by GpiSetClipRegion).

GpiEqualRegion – Equal Region

Related Functions

- GpiCombineRegion
- GpiCreateRegion
- GpiDestroyRegion
- GpiOffsetRegion
- GpiPaintRegion
- GpiPtInRegion
- GpiQueryRegionBox
- GpiQueryRegionRects
- GpiRectInRegion
- GpiSetRegion
- WinEqualRect

Example Code

This example uses GpiEqualRegion to create two regions (each consisting of three rectangles), and then compares them for equality.

```
#define INCL_GPIREGIONS      /* Region functions          */
#include <os2.h>

LONG    lEquality;          /* equality/error indicator    */
HPS      hps;               /* presentation-space handle   */
HRGN     hrgnSrc1;          /* handle for first region     */
HRGN     hrgnSrc2;          /* handle for second region    */
RECTL    arcl[3] = { 100, 100, 200, 200,          /* 1st rectangle              */
                     150, 150, 250, 250,          /* 2nd rectangle              */
                     200, 200, 300, 300 };         /* 3rd rectangle              */

/* create two identical regions comprising three rectangles each*/
hrgnSrc1 = GpiCreateRegion(hps, 3L, arcl);
hrgnSrc2 = GpiCreateRegion(hps, 3L, arcl);

lEquality = GpiEqualRegion(hps, hrgnSrc1, hrgnSrc2);
```

GpiErase — Erase

```
#define INCL_GPICONTROL /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

BOOL GpiErase (HPS hps)

This function clears the output display of the device context associated with the specified presentation space, to the reset color (CLR_BACKGROUND; see GpiSetColor).

Parameters

hps (HPS) — input
Presentation-space handle.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

Remarks

This operation is independent of the draw controls; see GpiSetDrawControl.

The call is subject to all clipping currently in force; that is, clip path, viewing limits, graphics field, clip region, and visible region.

This function does not perform any bounds collection, or correlation.

Note: This function must not be used when creating metafiles conforming to SAA* guidelines; see "Metafile Restrictions" on page G-1.

Related Functions

- GpiCreateLogColorTable
- GpiSetColor
- GpiSetDrawControl

Example Code

This example uses the GpiErase function to clear the display before drawing.

```
#define INCL_GPICONTROL      /* GPI control Functions      */
#include <os2.h>

HPS hps;                    /* presentation space handle */
POINTL ptlStart = { 0, 0 }; /* start point                */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices */

GpiErase(hps);              /* clear the display */
GpiMove(hps, &ptlStart);    /* draw a triangle    */
GpiPolyLine(hps, 3L, ptlTriangle);
```

GpiErrorSegmentData – Error Segment Data

```
#define INCL_GPICONTROL /* Or use INCL_GPI or INCL_PM */
```

LONG GpiErrorSegmentData (HPS hps, PLONG piSegment, PLONG piContext)

This function returns information about the last error that occurred during a segment drawing operation.

Parameters

hps (HPS) – input
Presentation-space handle.

piSegment (PLONG) – output
Segment in which the error occurred.

piContext (PLONG) – output
Context of the error:

GPIE_SEGMENT The error occurred while processing the contents of a retained segment.

GPIE_ELEMENT The error occurred while processing the contents of a GpiElement function.

GPIE_DATA The error occurred while processing the contents of a GpiPutData function.

Returns

Position.

This is either the byte offset or the element number, depending on *piContext*:

≥0 Position

GPI_ALTERERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION An attempt was made to issue a function that is invalid in a micro presentation space.

Remarks

The information returned is:

- The segment name
- The context
- The byte offset or element number (depending on the context).

The byte offset is returned for these contexts:

- The error occurred within the data of a GpiElement function
- The error occurred within the data of a GpiPutData function.

The element number is returned for the segment context.

Related Functions

- GpiElement
- GpiDrawChain
- GpiDrawDynamics
- GpiDrawFrom
- GpiDrawSegment
- GpiGetData
- GpiPutData
- GpiRemoveDynamics

Example Code

This example uses GpiErrorSegmentData to query the error context and assigns a variable to the returned element number if the context is an element error.

```
#define INCL_GPICONROL      /* Control functions      */
#include <os2.h>

LONG    lOff;              /* error or offset/element number */
HPS     hps;              /* presentation-space handle      */
LONG    plSegment;        /* Segment in which the error occurred */
LONG    plContext;        /* Context of the error           */
LONG    lElement;         /* element number causing error    */

lOff = GpiErrorSegmentData(hps, &plSegment, &plContext);

if (plContext == GPIE_ELEMENT)
    lElement = lOff;
```


GpiExcludeClipRectangle — Exclude Clip Rectangle

```
#define INCL_GPIREGIONS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiExcludeClipRectangle (HPS hps, PRECTL prclRectangle)

This function excludes a rectangle from the clipping region.

Parameters

hps (HPS) — input

Presentation-space handle.

prclRectangle (PRECTL) — input

Rectangle to be excluded.

The coordinates are world coordinates.

Returns

Complexity of clipping and error indicators.

The clipping complexity information includes the combined effects of:

- Clip path
- Viewing limits
- Graphics field
- Clip region
- Visible region (windowing considerations).

RGN_NULL Null region

RGN_RECT Rectangular region

RGN_COMPLEX Complex region

RGN_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE

An invalid coordinate value was specified.

PMERR_INV_RECT

An invalid rectangle parameter was specified.

Remarks

The boundaries of the rectangle are considered to be part of the interior, so that a point on the rectangle boundary is clipped (removed).

This function creates a clip region if one does not currently exist. The application is responsible for freeing this (with GpiDestroyRegion) if it subsequently selects another clip region (see GpiSetClipRegion). Any clip region still selected when the device context is closed is automatically freed.

Note: This function must not be used when creating SAA-conforming metafiles; see “Metafile Restrictions” on page G-1.

GpiExcludeClipRectangle — Exclude Clip Rectangle

Related Functions

- GpiIntersectClipRectangle
- GpiOffsetClipRegion
- GpiQueryClipBox
- GpiQueryClipRegion
- GpiSetClipRegion
- WinExcludeUpdateRegion

Example Code

This example uses GpiExcludeClipRectangle to exclude a 100x100 rectangle, anchored at (100,100), from the clipping region.

```
#define INCL_GPIREGIONS          /* Region functions          */
#include <os2.h>

LONG lComplexity;                /* clipping complexity/error return */
HPS hps;                        /* Presentation-space handle        */
RECTL prclRectangle = {100, 100, 200, 200}; /* exclude rectangle */

lComplexity = GpiExcludeClipRectangle(hps, &prclRectangle);
```

GpiFillPath — Fill Path

```
#define INCL_GPIPATHS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiFillPath (HPS hps, LONG IPath, LONG IOptions)

This function draws the interior of a path using the area attributes.

Parameters

hps (HPS) — input

Presentation-space handle.

IPath (LONG) — input

Identifier of path whose interior is to be drawn; it must be 1.

IOptions (LONG) — input

Fill option:

FPATH_ALTERNATE Fills the path using the alternate rule; see GpiBeginArea.

FPATH_WINDING Fills the path using the winding rule; see GpiBeginArea. This value must be selected if the path has been modified using GpiModifyPath.

The default is FPATH_ALTERNATE.

Returns

Error indicator:

GPI_OK Successful

GPI_HITS Correlate hits

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PATH_ID An invalid path identifier parameter was specified.

PMERR_INV_FILL_PATH_OPTIONS An invalid options parameter was specified with GpiFillPath.

PMERR_PATH_UNKNOWN An attempt was made to perform a path function on a path that did not exist.

Remarks

Any open figures within the path are closed.

The path is deleted when the interior has been drawn.

The boundaries of the area, as defined by the path, are considered to be part of the interior and are included in the fill.

If the current drawing mode (see GpiSetDrawingMode) is **draw** or **draw-and-retain**, the interior is drawn on the currently associated device. If the drawing mode is **retain**, this function is stored in the current segment, and output occurs when the segment is subsequently drawn in the usual way.

Related Functions

Prerequisite Functions

- GpiBeginPath

Other Related Functions

- GpiEndPath
- GpiModifyPath
- GpiOutlinePath
- GpiPathToRegion
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetClipPath
- GpiStrokePath
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom
- GpiSetPattern
- GpiSetPatternRefPoint
- GpiSetPatternSet
- GpiSetMix

Graphic Elements and Orders

Element Type: **OCODE_GFPTH**

Note that GpiStrokePath also generates this element type.

Order: **Fill Path**

Example Code

This example uses the GpiFillPath function to draw the interior of the given path. The path, an isosceles triangle, is not closed when it is created, so the GpiFillPath function closes it before filling.

```
#define INCL_GPIPATHS          /* GPI Path functions          */
#include <os2.h>

HPS hps;                      /* presentation space handle */
POINTL ptlStart = { 0, 0 }; /* first vertex                */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices */

GpiBeginPath(hps, 1L);        /* create a path */
GpiMove(hps, &ptlStart);
GpiPolyLine(hps, 3L, ptlTriangle);
GpiEndPath(hps);

GpiFillPath(hps, 1L, FPATH_ALTERNATE); /* fill the path */
```

GpiFloodFill — Flood Fill

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiFloodFill (HPS hps, LONG IOptions, LONG IColor)

This function fills an area bounded by a given color, or while on a given color.

Parameters

hps (HPS) — input
Presentation-space handle.

IOptions (LONG) — input
Flood fill options:

FF_BOUNDARY Fills up to the specified color

FF_SURFACE Fills while on the specified color.

IColor (LONG) — input
Color.

The boundary or surface color, depending on the value of *IOptions*.

This is either a logical color index, or an RGB value, depending on the state of the color table.

Returns

Correlation and error indicators:

GPI_OK Successful

GPI_HITS Correlate hits

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_FUNCTION_NOT_SUPPORTED	The function is not supported.
PMERR_INV_FLOOD_FILL_OPTIONS	Invalid flood fill parameters were specified.
PMERR_INV_IN_AREA	An attempt was made to issue a function invalid inside an area bracket. This can be detected while the actual drawing mode is draw or draw-and-retain or during segment drawing or correlation functions.
PMERR_INV_IN_PATH	An attempt was made to issue a function invalid inside a path bracket.
PMERR_INV_COLOR_ATTR	An invalid color attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INSUFFICIENT_MEMORY	The operation terminated through insufficient memory.
PMERR_START_POINT_CLIPPED	The starting point specified for flood fill is outside the current clipping path or region.

PMERR_NO_FILL

No flood fill occurred because either the starting point color was the same as the input color when a boundary fill was requested, or the starting point color was not the same as the input color when a surface fill was requested.

Remarks

The seed point is current position, which is unchanged by this function.

The area attributes define the fill.

DevQueryCaps (CAPS_RASTER_FLOOD_FILL) indicates whether GpiFloodFill is supported on any particular device.

The results produced by this function are highly device-dependent.

When the drawing mode is draw, if

If the presentation space is partially obscured by an overlying window an incorrect fill can result.

When filling over a pattern or a dithered color, the individual color of each pel is taken into account.

Note: This function must not be used when creating SAA-conforming metafiles; see "Metafile Restrictions" on page G-1.

Related Functions

Prerequisite Functions

- GpiBeginArea
- GpiBeginPath
- GpiFillPath
- GpiSetPel

Example Code

This function uses GpiFloodFill to fill an area bounded by a given color, or while on a given color. The example assumes the color table is in index mode; it fills up to the boundary where the color represented by index 1 appears.

```
#define INCL_GPIBITMAPS          /* Bit map functions          */
#include <os2.h>

LONG  lHits;                    /* correlation/error indicator */
HPS   hps;                      /* Presentation-space handle   */
LONG  lOptions;                 /* flood fill options          */
LONG  lColor;                   /* color                        */

/* fill up to the boundaries of the color */
lOptions = FF_BOUNDARY;

/* use color corresponding to index 1 */
lColor = 1;

lHits = GpiFloodFill(hps, lOptions, lColor);
```

GpiFrameRegion — Frame Region

```
#define INCL_GPIREGIONS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiFrameRegion (HPS hps, HRGN hrgn, PSIZEL pszIThickness)

This function draws a frame inside a region using the current pattern attributes.

Parameters

hps (HPS) — input
Presentation-space handle.

hrgn (HRGN) — input
Region handle.

pszIThickness (PSIZEL) — input
Thickness of frame.

The width and height of the rectangle, in device coordinates, used to trace the frame. Both the *width* and *height* fields must be greater than or equal to zero.

Returns

Correlation and error indicators:

GPI_OK Successful
GPI_HITS Correlate hits
GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_REGION_IS_CLIP_REGION	An attempt was made to perform a region operation on a region that is selected as a clip region.
PMERR_INV_HRGN	An invalid region handle was specified.
PMERR_HRGN_BUSY	An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

Remarks

The frame is drawn by tracing around the inner boundary of the region with a rectangle of size given by the *pszIThickness* parameter. The edge of the frame includes the pels on the left and bottom boundaries of the region, unless those pels are also on the top and right boundaries, in which case they are excluded.

No part of the frame is drawn outside the region.

The region is assumed to be defined in device coordinates.

It is invalid if the specified region is currently selected as the clip region (by GpiSetClipRegion).

Note: This function must not be used when creating SAA-conforming metafiles; see “Metafile Restrictions” on page G-1.

GpiFrameRegion — Frame Region

Example Code

This example uses GpiFrameRegion to draw a frame of width 5 around an existing region.

```
#define INCL_GPIREGIONS      /* Region functions      */
#include <os2.h>

LONG    lHits;              /* correlation/error indicator */
HPS     hps;                /* presentation-space handle   */
HRGN     hrgn;              /* handle for region           */
SIZEL    pszlThickness = {5L,5L};
                        /* Thickness of frame         */
RECTL arcl[3] = { 100, 100, 200, 200, /* 1st rectangle */
                  150, 150, 250, 250, /* 2nd rectangle */
                  200, 200, 300, 300 }; /* 3rd rectangle */

/* create a region comprising three rectangles */
hrgn = GpiCreateRegion(hps, 3L, arcl);

lHits = GpiFrameRegion(hps, hrgn, &pszlThickness);
```


GpiFullArc — Full Arc

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiFullArc (HPS hps, LONG IControl, FIXED fxMultiplier)

This function creates a full arc with its center at the current position.

Parameters

hps (HPS) — input
Presentation-space handle.

IControl (LONG) — input
Interior and outline control.

Specifies whether the interior of the full arc should be filled, and whether the outline should be drawn:

DRO_FILL	Fill interior
DRO_OUTLINE	Draw outline
DRO_OUTLINEFILL	Draw outline and fill interior.

fxMultiplier (FIXED) — input
Multiplier.

This determines the size of the arc, in relation to an arc with the current arc parameters. The implementation limit of the multiplier is 255.

The value must not be negative.

Returns

Correlation and error indicators:

GPI_OK	Successful
GPI_HITS	Correlate hits
GPI_ERROR	Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_ARC_CONTROL	An invalid control parameter was specified with GpiFullArc.
PMERR_INV_MULTIPLIER	An invalid multiplier parameter was specified with GpiPartialArc or GpiFullArc.

Remarks

The current position is not changed.

The arc parameters determine whether the full arc is drawn clockwise or counterclockwise.

Either the outline of the full arc, or its interior, or both, can be drawn.

If this function appears within an area or path definition, it generates a complete closed figure (DRO_OUTLINE must be specified). It must not occur within any other figure definition.

GpiFullArc – Full Arc

If correlation is in force, a hit always results if the pick aperture intersects the full arc boundary. However, if the pick aperture lies wholly within the figure, a hit only occurs if the interior is being drawn (DRO_FILL or DRO_OUTLINEFILL).

Related Functions

- GpiPartialArc
- GpiPointArc
- GpiSetCurrentPosition
- GpiSetArcParams
- GpiSetDefArcParams
- GpiSetLineType
- GpiSetLineWidth
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

Graphic Elements and Orders

Element Type: OCODE_GCFARC

Order: Begin Area

This order is generated only if *IControl* is DRO_FILL or DRO_OUTLINEFILL.

Order: Full Arc at Current Position

Order: End Area

This order is generated only if *IControl* is DRO_FILL or DRO_OUTLINEFILL.

Example Code

This example uses GpiFullArc to draw five concentric circles. The arc parameters are set before drawing the arc. Only the outline is drawn for the arc.

```
#define INCL_GPIPRIMITIVES    /* GPI primitive functions    */
#include <os2.h>

HPS hps;                    /* presentation space handle    */
SHORT i;                    /* loop variable                */
ARCPARAMS arcp = { 1, 1, 0, 0 }; /* arc parameters structure    */

GpiSetArcParams(hps, &arcp);

for (i = 5; i > 0; i--)
    GpiFullArc(hps,          /* presentation-space handle    */
               DRO_OUTLINE, /* outline                      */
               MAKEFIXED(i, 0)); /* converts integer to fixed point */
```

GpiGetData — Get Data

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiGetData (HPS hps, LONG ISegId, PLONG pOffset, LONG IFormat, LONG ILength, PBYTE pbData)

This function retrieves a buffer of graphic data from the specified segment into the supplied buffer. The data is a list of drawing orders. For details of these, see Chapter 33, “Graphics Orders.”

Parameters

hps (HPS) — input
Presentation-space handle.

ISegId (LONG) — input
Segment identifier.

pOffset (PLONG) — input/output
Segment offset.

A value used to indicate the position in the segment from which data is to be retrieved. It must be set to 0 the first time GpiGetData is called. This indicates that data is to be obtained from the start of the segment. On return, it contains a value that can be used on a subsequent call to continue data retrieval.

The only possible values that can be specified are 0 or the value returned from a previous function.

IFormat (LONG) — input
Coordinate type required:

DFORM_NOCONV No coordinate conversion performed.

ILength (LONG) — input
Length of data buffer.

pbData (PBYTE) — output
Data buffer.

For order formats, see Chapter 33, “Graphics Orders” on page 33-1.

Returns

Length of returned data.

≥0 Number of bytes actually returned in *pbData*

GPI_ALTERERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME An invalid segment identifier was specified.

PMERR_INV_SEG_OFFSET An invalid offset parameter was specified with GpiPutData.

PMERR_INV_GETDATA_CONTROL An invalid format parameter was specified with GpiGetData.

PMERR_INV_LENGTH_OR_COUNT An invalid length or count parameter was specified.

PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.
PMERR_SEG_NOT_FOUND	The specified segment identifier did not exist
PMERR_SEG_IS_CURRENT	An attempt was made to issue GpiGetData to a segment that was currently open.
PMERR_DATA_TOO_LONG	An attempt was made to transfer more than the maximum permitted amount of data (64512 bytes) using GpiPutData, GpiGetData, or GpiElement.

Remarks

If the buffer is large enough to contain the data requested, the data is returned and *ICount* is set to show its length.

If the buffer is not large enough, the buffer is filled and *ICount* is set to the length of the buffer. This may mean that there is an incomplete order at the end of the buffer; even so, it is possible to use GpiPutData subsequently, without having to scan the orders in the buffer.

The application can detect when it has been given all the data by checking the *ICount* value. If this is less than the value of *ILength* specified, there is no more data to be returned. If it is equal, there is more data, except in the case where the data just fits in the buffer, which is detected if another GpiGetData function is issued, and a *ICount* of 0 is returned.

No conversion of coordinates is performed for the DFORM_NOCONV value of the control parameter. The coordinates are in the presentation space format.

This function can be issued while there is a segment open, unless the open segment is the segment referenced by this function. If the segment referenced by this function is open, an error occurs.

The segment transform and viewing transform are not returned by this call.

Related Functions

- GpiPutData

GpiGetData —

Get Data

Example Code

This example uses the GpiGetData function to copy data from one segment to another.

```
#define INCL_GPISEGMENTS      /* Segment functions          */
#include <os2.h>

HPS hps;                      /* presentation space handle */
LONG fFormat = DFORM_NOCONV; /* does not convert coordinates */
LONG offSegment = 0L;         /* offset in segment          */
LONG offNextElement = 0L;     /* offset in segment to next element */
LONG cb = 0L;                 /* bytes retrieved            */
BYTE abBuffer[512];           /* data buffer                 */

GpiOpenSegment(hps, 3L);      /* opens segment to receive data */
do {
    offSegment += cb;
    offNextElement = offSegment;
    cb = GpiGetData(hps, 2L, &offNextElement, fFormat, 512L,
                    abBuffer);

    /* Put data in other segment. */

    if (cb > 0L) GpiPutData(hps, /* presentation-space handle */
                            fFormat, /* format of coordinates */
                            &cb, /* number of bytes in buffer */
                            abBuffer); /* buffer with graphics-order data */

} while (cb > 0);
GpiCloseSegment(hps);         /* closes segment that received data */
```

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG Gpimage (HPS hps, LONG IFormat, PSIZEL pszImageSize, LONG ILength, PBYTE pbData)

This function draws a rectangular image, with the top-left corner at the current position.

Parameters

- hps (HPS)** – input
Presentation-space handle.
- IFormat (LONG)** – input
Format of image data.
This is a reserved field; must be set to 0.
- pszImageSize (PSIZEL)** – input
Size of image area (in pels).
The maximum width allowed is 2 040.
- ILength (LONG)** – input
Length in bytes of image data.
- pbData (PBYTE)** – input
Image data.

Returns

Correlation and error indicators:

- GPI_OK** Successful
- GPI_HITS** Correlate hits
- GPI_ERROR** Error.

Possible returns from WinGetLastError

- | | |
|------------------------------------|---|
| PMERR_INV_HPS | An invalid presentation-space handle was specified. |
| PMERR_PS_BUSY | An attempt was made to access the presentation space from more than one thread simultaneously. |
| PMERR_INV_IMAGE_FORMAT | An invalid <i>IFormat</i> parameter was specified with Gpimage. |
| PMERR_INV_IMAGE_DATA_LENGTH | An invalid <i>ILength</i> parameter was specified with Gpimage. There is a mismatch between the image size and the data length. |
| PMERR_INV_IMAGE_DIMENSION | An invalid <i>pszImageSize</i> parameter was specified with Gpimage. |

Remarks

All images are a rectangular array of pels (display points), each pel being represented by one bit.

pszImageSize, which defines the width and height of the image, determines how many pels there are in the horizontal and vertical directions.

GpImage —

Image

pbData determines which of the pels are visible; a 1 bit sets the associated pel, using the image foreground color and mix, and a 0 bit sets the pel using the image background color and mix.

The top left-hand corner of the image is placed at the current position, and the data supplied is drawn row by row, starting at the top. Each row is drawn from left to right and must be padded out to an integral number of bytes if the image width specified is not a multiple of 8. For example, if the image width specified is 12, each row of data must be padded out to a length of 16 so that the data in the row occupies exactly 2 bytes.

Within each byte the high-order bit is drawn on the left.

The length of image data specified must include the padding of each row of data. The length must be given in bytes, and an error message is issued if it is wrong.

If the image is being stored in a metafile, then $((\text{pels_per_row} + 9) / 8) * \text{pels_per_column} + 10$, must be less than 32768.

Because of the different sizes of pels for different devices, the relationship of the image with respect to other graphics primitives is device-dependent.

The current position remains unchanged after the image has been drawn.

Related Functions

- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

Graphic Elements and Orders

Element Type: **OCODE_GCBIMG**

Order: **Begin Image at Current Position**

Order: **Image Data**

One order for each pel row of the image.

Order: **End Image**

Example Code

This example uses GpImage to draw an 8-by-8 image. The image data is specified as an array of bytes.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions */
#include <os2.h>

HPS hps;                      /* presentation space handle */
SIZEL sizl = { 8, 8 }; /* image is 8 pels wide by 8 pels high */
BYTE abImage[] = { 0x00, 0x18, 0x3c, 0x7e, 0xff,
                   0xff, 0x7e, 0x3c, 0x18, 0x00 }; /* image data */

GpiImage(hps, 0L, &sizl, 8L, abImage); /* draws the image */
```

GpIntersectClipRectangle – Intersect Clip Rectangle

```
#define INCL_GPIREGIONS /* Or use INCL_GPI or INCL_PM */
```

LONG GpIntersectClipRectangle (HPS hps, PRECTL prclRectangle)

This function sets the new clip region to the intersection of the current clip region and the specified rectangle.

Parameters

hps (HPS) – input

Presentation-space handle.

prclRectangle (PRECTL) – input

prclRectangle, the coordinates of which are world coordinates.

Returns

Complexity of clipping and error indicators.

The clipping complexity information includes the combined effects of:

- Clip path
- Viewing limits
- Graphics field
- Clip region
- Visible region (windowing considerations).

RGN_NULL Null region

RGN_RECT Rectangular region

RGN_COMPLEX Complex region.

RGN_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE

An invalid coordinate value was specified.

PMERR_INV_RECT

An invalid rectangle parameter was specified.

Remarks

The boundaries of the rectangle are considered to be part of the interior, so that a point on the rectangle boundary is not clipped (removed) if it was previously within the clip region.

This function creates a clip region if one does not currently exist. The application is responsible for freeing this (with GpiDestroyRegion), if it subsequently selects another clip region (see GpiSetClipRegion). Any clip region still selected when the device context is closed is automatically freed.

Note: This function must not be used when creating SAA-conforming metafiles; see “Metafile Restrictions” on page G-1.

GpiIntersectClipRectangle — Intersect Clip Rectangle

Related Functions

- GpiExcludeClipRectangle
- GpiOffsetClipRegion
- GpiQueryClipBox
- GpiQueryClipRegion
- GpiSetClipRegion

Example Code

This example uses GpiIntersectClipRectangle to create a new clipping region, consisting of the intersection of the old clipping region and a 100x100 rectangle, anchored at (100,100).

```
#define INCL_GPIREGIONS      /* Region functions      */
#include <os2.h>

LONG lComplexity;           /* clipping complexity/error return */
HPS hps;                   /* Presentation-space handle      */
RECTL prclRectangle = {100,100,200,200}; /* intersect rectangle */

lComplexity = GpiIntersectClipRectangle(hps, &prclRectangle);
```

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiLabel (HPS hps, LONG ILabel)

This function generates an element containing the specified label.

Parameters

hps (HPS) – input
Presentation-space handle.

ILabel (LONG) – input
Required label.

No check is made on the value of this parameter.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION

An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_IN_ELEMENT

An attempt was made to issue a function invalid inside an element bracket.

Remarks

This function has no effect unless a retained segment is being constructed. It is invalid within an element bracket. Duplicate labels within a segment are allowed.

Related Functions

- GpiSetElementPointerAtLabel
- GpiSetTag

Graphic Elements and Orders

Element Type: **OCODE_GLABL**

Order: **Label**

GpiLabel — Label

Example Code

This example uses the GpiLabel function to create label elements in a segment. If the segment is subsequently edited, the label elements can still be used to locate the elements near it.

```
#define INCL_GPISEGEDITING    /* GPI Segment Edit functions */
#include <os2.h>

HPS hps;                    /* presentation space handle */
POINTL ptlStart = { 0, 0 }; /* first vertex */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices */

GpiOpenSegment(hps, 4L);    /* creates a segment */
GpiLabel(hps, 5L);         /* creates label 5 */
GpiLabel(hps, 10L);        /* creates label 10 */
GpiMove(hps, &ptlStart);
GpiCloseSegment(hps);
GpiPolyLine(hps, 3L, ptlTriangle);
```

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

LONG GpiLine (HPS hps, PPOINTL pptlEndPoint)

This function draws a straight line from the current position to the specified end point.

Parameters

hps (HPS) — input
Presentation-space handle.

pptlEndPoint (PPOINTL) — input
End point of the line.

Returns

Correlation and error indicators:

GPI_OK Successful

GPI_HITS Correlate hits

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.
PMERR_INV_NESTED_FIGURES	Nested figures have been detected within a path definition.

Remarks

The current position is set to the end point of the line.

The line is drawn using the current values of the line color, line mix, line width, and line type attributes.

GpiLine — Line

Related Functions

- GpiBox
- GpiMove
- GpiPolyLine
- GpiQueryCurrentPosition
- GpiSetCurrentPosition
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

Graphic Elements and Orders

Element Type: **OCODE_GCLINE**

Note that GpiPolyLine also generates this element type.

Order: **Line at Current Position**

Example Code

This example uses GpiLine to draw an X.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions */
#include <os2.h>

HPS hps;                        /* presentation space handle */
/* point array */
POINTL ptl[4] = { 0, 0, 100, 100, 0, 100, 100, 0 };

GpiMove(hps, &ptl[0]);
GpiLine(hps, &ptl[1]);
GpiMove(hps, &ptl[2]);
GpiLine(hps, &ptl[3]);
```

GpiLoadBitmap – Load Bit Map

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

HBITMAP GpiLoadBitmap (HPS hps, HMODULE Resource, ULONG IdBitmap, LONG IWidth, LONG IHeight)

This function creates and loads a bit map from a resource, and returns the bit-map handle.

Parameters

hps (HPS) – input

Presentation-space handle.

The associated device should, if possible, hold the bit map in its own memory. Where this is not possible, main memory is used and the bit map is held in a format compatible with the device.

Resource (HMODULE) – input

Resource identity containing the bit map:

NULLHANDLE Use the .EXE file of the application.

Other Module handle returned from the OS/2 DosLoadModule function.

IdBitmap (ULONG) – input

ID of the bit map within the resource file.

IWidth (LONG) – input

Width of the bit map in pels.

IHeight (LONG) – input

Height of the bit map in pels.

Returns

Bit-map handle:

≠0 Bit-map handle

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_BITMAP_NOT_FOUND

A attempt was made to perform a bit-map operation on a bit map that did not exist.

PMERR_INV_BITMAP_DIMENSION

An invalid dimension was specified with a load bit-map function.

Remarks

Some bit-map functions, including drawing into the bit map, require it to be selected into a memory device context, using GpiSetBitmap. This is true whether device or main memory is used to hold the bit map.

The bit map is stretched to the specified *IWidth* and *IHeight*. If *IWidth* or *IHeight* is 0, the bit map is not stretched in that direction; when, for example, *IWidth* = 0, the bit map is not stretched horizontally, when *IHeight* = 0, it is not stretched vertically.

The bit map may have been created by the icon editor in bit-map mode.

GpiLoadBitmap —

Load Bit Map

There are a number of standard bit-map formats that should normally be adhered to. Other formats can be used if supported by the device.

The bit map is owned by the process from which this function is issued. It cannot be accessed directly from any other process. If it still exists when the process terminates, it is automatically deleted by the system.

Related Functions

- GpiBitBlt
- GpiCreateBitmap
- GpiDeleteBitmap
- GpiDrawBits
- GpiQueryBitmapBits
- GpiQueryBitmapDimension
- GpiQueryBitmapHandle
- GpiQueryBitmapParameters
- GpiQueryDeviceBitmapFormats
- GpiSetBitmap
- GpiSetBitmapBits
- GpiSetBitmapDimension
- GpiSetBitmapId
- GpiWCBitBlt
- WinDrawBitmap
- WinGetSysBitmap

Example Code

This example uses the GpiLoadBitmap function to load a bit map from the .EXE file into application memory. The bit map is then selected, displayed, and finally, deleted from memory.

```
#define INCL_GPIBITMAPS          /* GPI bit map functions      */
#include <os2.h>

HPS hps;                        /* presentation space handle */
HBITMAP hbm, hbmPrevious;
#define BITMAP_ID 1

/* load the bit map from the EXE */
hbm = GpiLoadBitmap(hps, NULLHANDLE, BITMAP_ID, 100L, 100L);
hbmPrevious = GpiSetBitmap(hps, hbm); /* select bit map for PS */

/* bit map displayed with GpiBitBlt */

GpiSetBitmap(hps, hbmPrevious);    /* release bit map from PS */
GpiDeleteBitmap(hbm);              /* delete the bit map      */
```

GpiLoadFonts – Load Fonts

```
#define INCL_GPILCIDS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiLoadFonts (HAB hab, PSZ pszFilename)

This function loads one or more fonts from the specified resource file.

Parameters

hab (HAB) – input
Anchor-block handle.

pszFilename (PSZ) – input
Filename.

This is the fully-qualified name of the font resource. The file-name extension is ".FON."

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from GetLastError

PMERR_INV_FONT_FILE_DATA

The font file specified with GpiLoadFonts,
GpiLoadPublicFonts,

Remarks

All of the fonts in the file become available for any presentation space (GPI or VIO) created by the same process. They are not available for any other process.

The format of the font definitions in the resource file is defined in Appendix F, "The Font-File Format" on page F-1.

When no longer required, the fonts may be unloaded with GpiUnloadFonts.

Note: Fonts loaded with GpiLoadFonts are not available for use for spooled printing, that is if a device type of OD_QUEUED is specified in DevOpenDC; in this case GpiCreateLogFont will never return FONT_MATCH for these fonts. To avoid this, install the fonts as public fonts using the Font Palette object located in the System Setup folder, on both the generating and the receiving workstations if these are different.

Related Functions

- GpiCreateLogFont
- GpiDeleteSetId
- GpiQueryFontMetrics
- GpiQueryFonts
- GpiQueryKerningPairs
- GpiQueryNumberSetIds
- GpiQuerySetIds
- GpiQueryWidthTable
- GpiUnloadFonts
- GpiSetCharSet

GpiLoadFonts — Load Fonts

Example Code

This example uses the GpiLoadFonts function to load all fonts from the font resource file HELV.FON. The GpiQueryFonts function retrieves the number of fonts loaded.

```
#define INCL_GPILCIDS          /* Font functions          */
#include <os2.h>

HPS hps;                      /* presentation space handle */
HAB hab;                      /* anchor-block handle       */
LONG cFonts = 0L;            /* font count                 */
LONG remFonts;               /* fonts not returned         */

GpiLoadFonts(hab, "helv");

remFonts = GpiQueryFonts(hps, QF_PRIVATE, NULL, &cFonts, 0L, NULL);
```

GpiLoadMetaFile – Load Metafile

```
#define INCL_GPIMETAFILES /* Or use INCL_GPI or INCL_PM */
```

HMF GpiLoadMetaFile (HAB hab, PSZ pszFilename)

This function loads data from a file into a metafile.

Parameters

hab (HAB) – input

Anchor-block handle.

pszFilename (PSZ) – input

Filename.

The name of the file that is to be loaded into a metafile.

Returns

Metafile handle or error:

≠0 Metafile handle

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_DOSOPEN_FAILURE

A DosOpen call made during GpiLoadMetaFile or GpiSaveMetaFile gave a good return code but the file was not opened successfully.

PMERR_DOSREAD_FAILURE

A DosRead call made during GpiLoadMetaFile gave a good return code. However, it failed to read any more bytes although the file length indicated that there were more to be read.

Remarks

A metafile is created, into which the data from the file is loaded. The handle of the metafile created is returned in *hmf*; it can be used on subsequent GpiPlayMetaFile or GpiDeleteMetaFile functions.

Related Functions

- GpiCopyMetaFile
- GpiDeleteMetaFile
- GpiPlayMetaFile
- GpiQueryMetaFileBits
- GpiQueryMetaFileLength
- GpiSaveMetaFile
- GpiSetMetaFileBits

GpiLoadMetaFile — Load Metafile

Example Code

This example uses the GpiLoadMetaFile function to load a metafile with data from the file sample.met. Later, the metafile is deleted by using the GpiDeleteMetaFile function.

```
#define INCL_GPIMETAFILES      /* Metafile functions      */
#include <os2.h>

HAB hab;                      /* anchor block handle */
HMF hmf;                      /* metafile handle      */

/* loads metafile from disk */
hmf = GpiLoadMetaFile(hab, "sample.met");
.
.
.
GpiDeleteMetaFile(hmf);      /* deletes metafile     */
```

GpiLoadPublicFonts – Load Public Fonts

```
#define INCL_GPILCIDS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiLoadPublicFonts (HAB hab, PSZ pszFilename)

This function loads one or more fonts from the specified resource file, to be available for all applications.

Parameters

hab (HAB) – input
Anchor-block handle.

pszFilename (PSZ) – input
Filename.

This is the fully-qualified name of the font resource. The file-name extension is ".FON."

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INSUFFICIENT_MEMORY The operation terminated through insufficient memory.

PMERR_INV_FONT_FILE_DATA The font file specified with GpiLoadFonts,
GpiLoadPublicFonts,

Remarks

All of the fonts in the file become available for any presentation space (GPI or VIO) created by any process.

The format of the font definitions in the resource file is defined in Appendix F, "The Font-File Format" on page F-1.

Note: Problems can occur when applications load and unload public fonts. See
GpiUnloadPublicFonts.

Example Code

This example use GpiLoadPublicFonts to load and make available fonts from a file 'TEST.FON', which is assumed to exist and contain valid fonts.

```
#define INCL_GPILCIDS          /* Font functions          */
#include <os2.h>

BOOL fSuccess;                /* success indicator          */
HAB hab;                      /* anchor-block handle        */
char pszFilename[13];         /* Name of fond resource file */

/* resource file is named 'TEST.FON' */
strcpy(pszFilename,"TEST.FON");

fSuccess = GpiLoadPublicFonts(hab, pszFilename);
```

GpiMarker — Marker

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiMarker (HPS hps, PPOINTL pptlPoint)

This function draws a marker with its center at a specified position.

Parameters

hps (HPS) — input
Presentation-space handle.

pptlPoint (PPOINTL) — input
Position of the marker.

Returns

Correlation and error indicators:

GPI_OK Successful
GPI_HITS Correlate hits
GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.

Remarks

The current position is moved to the specified position. The marker symbol is selected by the current values of the marker set and marker symbol attributes.

Related Functions

- GpiPolyMarker
- GpiSetMarker
- GpiSetMarkerBox
- GpiSetMarkerSet
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

Graphic Elements and Orders

Element Type: **OCODE_GMRK**

Note that GpiPolyMarker also generates this element type.

Order: **Marker at Given Position**

Example Code

This example uses the GpiMarker function to draw a marker at the point (10,10).

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                        /* presentation space handle */
POINTL pt1 = { 10, 10 }; /* marker point */

GpiMarker(hps, &pt1);
```

GpiModifyPath — Modify Path

```
#define INCL_GPIPATHS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiModifyPath (HPS hps, LONG IPath, LONG IMode)

This function modifies the specified path.

Parameters

hps (HPS) — input

Presentation-space handle.

IPath (LONG) — input

Path identifier.

Identifier of the path to be modified; it must be 1.

IMode (LONG) — input

Modification required.

This must be:

MPATH_STROKE Convert the path to one describing the envelope of a wide line.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from GetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PATH_ID

An invalid path identifier parameter was specified.

PMERR_INV_MODIFY_PATH_MODE

An invalid mode parameter was specified with GpiModifyPath.

PMERR_PATH_UNKNOWN

An attempt was made to perform a path function on a path that did not exist.

PMERR_COORDINATE_OVERFLOW

An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

Remarks

This function converts the path to one describing the envelope of a wide line stroked using the current geometric wide-line attribute (see GpiSetLineWidthGeom). Note that this and GpiStrokePath are the only calls that can cause geometric wide lines to be constructed.

The envelope includes the effects of line joins, and line ends, according to the current values of these attributes (see GpiSetLineJoin and GpiSetLineEnd). Note these points:

- A line may be joined to an arc, for example. The common point is handled according to the line-join attribute, rather than applying line ends at each end.
- Any open figures within the path are not closed automatically.

GpiModifyPath – Modify Path

- If a figure is closed using GpiCloseFigure, the joining rules are followed, rather than the ending rules, at the start and end point.
- The envelope takes account of any crossings, so that a character such as a stroked "X" does not have a hole in the middle when subsequently drawn in exclusive-OR mode.

After this function, the only calls that can be performed on the path are GpiFillPath, specifying the FPATH_WINDING option, or GpiSetClipPath, specifying the SCP_WINDING option.

Related Functions

- GpiBeginPath
- GpiEndPath
- GpiFillPath
- GpiOutlinePath
- GpiPathToRegion
- GpiSetClipPath
- GpiSetPattern
- GpiSetPatternRefPoint
- GpiSetPatternSet
- GpiStrokePath
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

Graphic Elements and Orders

Element Type: OCODE_GMPH

Order: Modify Path

GpiModifyPath — Modify Path

Example Code

This example uses the GpiModifyPath function to modify the given path. The GpiFillPath function then draws the path.

```
#define INCL_GPIPATHS          /* GPI Path functions */
#include <os2.h>

HPS hps;                      /* presentation space handle */
POINTL ptlStart = { 0, 0 }; /* first vertex */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices */

GpiBeginPath(hps, 1L);        /* creates path */
GpiMove(hps, &ptlStart);
GpiPolyLine(hps, 3L, ptlTriangle);
GpiEndPath(hps);

GpiModifyPath(hps,
              1L,
              MPATH_STROKE); /* modifies path for wide line */
GpiFillPath(hps, 1L, FPATH_ALTERNATE); /* draws the wide line */
```

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

BOOL GpiMove (HPS hps, PPOINTL pptlPoint)

This function moves the current position to the specified point.

Parameters

hps (HPS) – input
Presentation-space handle.

pptlPoint (PPOINTL) – input
Position to which to move.
This position is in world coordinates.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.

Remarks

This function also has the effect of resetting position within a line-type sequence, and, if within an area, of starting a new closed figure and causing any previous one to be closed automatically if necessary.

This function is equivalent to the GpiSetCurrentPosition call, except that, if the current attribute mode is AM_PRESERVE (see GpiSetAttrMode), the current position is **not** saved before being set to a new value by the GpiMove function, and hence cannot be restored using the GpiPop call.

Related Functions

- GpiQueryCurrentPosition
- GpiSetCurrentPosition

Graphic Elements and Orders

Element Type: **OCODE_GSCP**

Note that GpiSetCurrentPosition also generates this element type.

Order: **Set Current Position**

GpiMove — Move

Example Code

This example uses the GpiMove function to draw an X. The function moves the current position to the starting point of each leg of the character.

```
#define INCL_GPIPRIMITIVES    /* GPI primitive functions */
#include <os2.h>

HPS hps;                    /* presentation space handle */
/* point array */
POINTL ptl[4] = { 0, 0, 100, 100, 0, 100, 100, 0 };

GpiMove(hps, &ptl[0]);      /* move to (0,0) */
GpiLine(hps, &ptl[1]);
GpiMove(hps, &ptl[2]);      /* move to (0,100) */
GpiLine(hps, &ptl[3]);
```

GpiOffsetClipRegion – Offset Clip Region

```
#define INCL_GPIREGIONS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiOffsetClipRegion (HPS hps, PPOINTL pptiPoint)

This function moves the clipping region by the specified displacement.

Parameters

hps (HPS) – input
Presentation-space handle.

pptiPoint (PPOINTL) – input
Displacement.

The displacement by which the clipping region is to be moved, expressed as an offset in world coordinates.

Returns

Complexity of clipping and error indicators.

The clipping complexity information includes the combined effects of:

- Clip path
- Viewing limits
- Graphics field
- Clip region
- Visible region (windowing considerations).

RGN_NULL	Null region
RGN_RECT	Rectangular region
RGN_COMPLEX	Complex region
RGN_ERROR	Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_COORDINATE_OVERFLOW	An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

Remarks

Note: This function must not be used when creating SAA-conforming metafiles; see “Metafile Restrictions” on page G-1.

Related Functions

- GpiExcludeClipRectangle
- GpiIntersectClipRectangle
- GpiQueryClipBox
- GpiQueryClipRegion
- GpiSetClipRegion
- WinExcludeUpdateRegion

GpiOffsetClipRegion — Offset Clip Region

Example Code

This example uses GpiOffsetClipRegion to move the clipping region right by 3 and up by 3.

```
#define INCL_GPIREGIONS      /* Region functions      */
#include <os2.h>

LONG lComplexity;           /* clipping complexity/error return */
HPS hps;                    /* Presentation-space handle        */
POINTL pptlPoint = {3,3}; /* displacement                      */

lComplexity = GpiOffsetClipRegion(hps, &pptlPoint);
```

GpiOffsetElementPointer – Offset Element Pointer

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiOffsetElementPointer (HPS hps, LONG loffset)

This function sets the element pointer, within the current segment, to the current value plus the specified offset.

Parameters

hps (HPS) – input
Presentation-space handle.

loffset (LONG) – input
Offset to be added to the element pointer.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.
PMERR_NOT_IN_RETAIN_MODE	An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to retain
PMERR_NO_CURRENT_SEG	An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open segment.
PMERR_INV_IN_ELEMENT	An attempt was made to issue a function invalid inside an element bracket.

Remarks

If the resulting value is negative, the element pointer is set to 0. If the resulting value is greater than the number of elements in the segment, it is set to the last element.

This function is only valid when the drawing mode (see GpiSetDrawingMode) is set to **retain** (not **draw-and-retain**), and a segment bracket is currently in progress.

This function is invalid within an element bracket.

GpiOffsetElementPointer – Offset Element Pointer

Related Functions

- GpiBeginElement
- GpiDeleteElement
- GpiDeleteElementRange
- GpiDeleteElementsBetweenLabels
- GpiElement
- GpiEndElement
- GpiLabel
- GpiQueryElement
- GpiQueryElementPointer
- GpiQueryElementType
- GpiSetElementPointer
- GpiSetElementPointerAtLabel

Example Code

This example uses the GpiOffsetElementPointer function to move to the element associated with a label element. Combining the GpiSetElementPointerAtLabel and GpiOffsetElementPointer functions is a convenient way to locate elements in segments that have been edited.

```
#define INCL_GPISEGEDITING      /* GPI Segment Edit functions */
#define INCL_GPISEGMENTS       /* Segment functions */
#include <os2.h>

HPS hps;          /* presentation space handle */
POINTL ptlStart = { 0, 0 }; /* first vertex */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices */

GpiOpenSegment(hps, 4L);          /* creates a segment with labels */
GpiLabel(hps, 5L);  GpiMove(hps, &ptlStart);
GpiLabel(hps, 10L); GpiPolyLine(hps, 3L, ptlTriangle);
GpiCloseSegment(hps);

.
.
.

GpiOpenSegment(hps, 4L);
GpiSetElementPointerAtLabel(hps, 10L); /* move to label 10 */
GpiOffsetElementPointer(hps, 1L);     /* move to polyline element */
```

GpiOffsetRegion – Offset Region

```
#define INCL_GPIREGIONS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiOffsetRegion (HPS hps, HRGN Hrgn, PPOINTL pptlOffset)

This function moves a region.

Parameters

hps (HPS) – input

Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

Hrgn (HRGN) – input

Handle of the region to be moved.

pptlOffset (PPOINTL) – input

Offset to be added to the region boundary.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from GetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HRGN

An invalid region handle was specified.

PMERR_REGION_IS_CLIP_REGION

An attempt was made to perform a region operation on a region that is selected as a clip region.

PMERR_INV_COORDINATE

An invalid coordinate value was specified.

PMERR_HRGN_BUSY

An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

Remarks

This function moves the region to a new position. The new position is obtained by adding the value of *pptlOffset* to all the points that define the region boundary.

An error is raised if the specified region is currently selected as the clip region (by *GpiSetClipRegion*).

GpiOffsetRegion — Offset Region

Related Functions

- GpiCombineRegion
- GpiCreateRegion
- GpiDestroyRegion
- GpiEqualRegion
- GpiPaintRegion
- GpiPtInRegion
- GpiQueryRegionBox
- GpiQueryRegionRects
- GpiRectInRegion
- GpiSetRegion

Example Code

This example uses GpiOffsetRegion to move a region right by 3 and up by 3.

```
#define INCL_GPIREGIONS      /* Region functions          */
#include <os2.h>

BOOL  fSuccess;              /* success indicator          */
HPS    hps;                  /* Presentation-space handle  */
HRGN   Hrgn;                 /* handle for region          */
POINTL pptlOffset = {3,3}; /* displacement                */

fSuccess = GpiOffsetRegion(hps, Hrgn, &pptlOffset);
```

GpiOpenSegment – Open Segment

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiOpenSegment (HPS hps, LONG ISegment)

This function opens a segment with the specified identification number.

Parameters

hps (HPS) – input
Presentation-space handle.

ISegment (LONG) – input
Segment identifier.
Must be zero or a positive number.

Returns

Success indicator:

TRUE Successful completion
FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME

An invalid segment identifier was specified.

PMERR_INV_MICROPS_FUNCTION

An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_ALREADY_IN_SEG

An attempt was made to open a new segment while an existing segment bracket was already open.

PMERR_PATH_INCOMPLETE

An attempt was made to open or close a segment either directly or during segment drawing, or to issue GpiAssociate while there is an open path bracket.

PMERR_AREA_INCOMPLETE

Either:

- A segment has been opened, closed, or drawn.
- GpiAssociate was issued while an area bracket was open.
- A drawn segment has opened an area bracket and ended without closing it.

PMERR_INV_MODE_FOR_REOPEN_SEG

An attempt was made to reopen an existing segment while the drawing mode was set to DM_DRAW or DM_DRAWANDRETAIN.

PMERR_DYNAMIC_SEG_ZERO_INV

An attempt was been made to open a dynamic segment with a segment identifier of zero.

PMERR_INV_MODE_FOR_OPEN_DYN

An attempt was made to open a segment with the ATTR_DYNAMIC segment set, while the drawing mode was set to DM_DRAW or DM_DRAWANDRETAIN.

GpiOpenSegment —

Open Segment

PMERR_UNCHAINED_SEG_ZERO_INV

An attempt was made to open segment with segment identifier zero and the ATTR_CHAINED segment attribute not specified.

Remarks

A segment is a way of grouping graphics primitives.

If the current drawing mode is **retain** or **draw-and-retain** (see GpiSetDrawingMode), the following occurs:

- If a nonzero identifier is given, and if a segment with the specified identifier does not already exist, a new retained segment is created. If one does already exist, it is reopened in **retain** mode (with the element pointer set to 0), but is an error in **draw-and-retain** mode.
- If an identifier of 0 is given, a new retained segment is created, regardless of whether one with a 0 identifier already exists. There can be more than one segment with an identifier of 0, but such segments can never subsequently be referenced by identifier. When they have been created, they continue to exist until all segments are deleted. Zero segments must be chained and cannot be defined as dynamic.

If the current drawing mode is **draw**, a new nonretained segment is started. No check is made against any possible retained segment identifiers. The current attributes are set to default values (subject to the ATTR_FASTCHAIN segment attribute; see below).

The initial attributes of the segment are as set by GpiSetInitialSegmentAttrs. The attributes may subsequently be changed with GpiSetSegmentAttrs (except for a segment with an identifier of 0). It is an error to try to open a new segment with a drawing mode of **draw** or **draw-and-retain**, with the ATTR_DYNAMIC segment attribute.

This function causes a segment bracket to be started. While the bracket is in effect, any primitive and attribute functions are considered to be part of the segment, and are stored in it if the drawing mode is **retain** or **draw-and-retain**. The bracket is terminated by a GpiCloseSegment. It is an error if GpiOpenSegment is issued when a segment is already open.

The following actions occur when drawing of a chained segment is started (either as it is passed across the API in **draw** or **draw-and-retain**, or as it is found during a draw operation):

- Current attributes and arc parameters are reset to default values.
- The current tag is reset to its default value.
- Current model transform is reset to unity.
- Current position is set to (0,0).
- The current clip path is set so as to cause no clipping.
- The current viewing limits are reset to their default values.
- The current viewing transform is set either to the value last set by GpiSetViewingTransformMatrix, or to the default value if no GpiSetViewingTransformMatrix function has been issued.

If the segment has the ATTR_FASTCHAIN attribute, the application should not depend upon whether or not these operations are performed. This avoids complications when interchanging picture data with other implementations.

Note: The current clip region is not changed by this function.

If any primitive/attribute calls are issued immediately before this function (that is, outside a segment bracket), then any currently open area, path, or element brackets are terminated, as described for GpiCloseSegment, before the new segment is opened.

If the segment being defined is to be called from another segment (see GpiCallSegmentMatrix), ensure that the viewing transform (see GpiSetViewingTransformMatrix) is unity before first opening the segment.

GpiOpenSegment — Open Segment

The maximum number of retained segments allowed for a given presentation space at any time is 16378.

Related Functions

- GpiCallSegmentMatrix
- GpiCloseSegment
- GpiCorrelateSegment
- GpiDeleteSegment
- GpiDeleteSegments
- GpiDrawSegment
- GpiErrorSegmentData
- GpiQueryInitialSegmentAttrs
- GpiQuerySegmentAttrs
- GpiQuerySegmentNames
- GpiQuerySegmentPriority
- GpiSetInitialSegmentAttrs
- GpiSetSegmentAttrs
- GpiSetSegmentPriority
- GpiSetViewingTransformMatrix

Example Code

This example uses the GpiOpenSegment to create a new segment. The segment is subsequently drawn by using the GpiDrawSegment function.

```
#define INCL_GPISEGMENTS      /* Segment functions      */
#include <os2.h>

HPS hps;                      /* presentation space handle */
POINTL ptlStart = { 0, 0 }; /* first vertex */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices */

GpiOpenSegment(hps, 1L);      /* opens the segment */
GpiMove(hps, &ptlStart);      /* moves to starting point (0,0) */
GpiPolyLine(hps, 3L, ptlTriangle); /* draws triangle */
GpiCloseSegment(hps);         /* closes the segment */

GpiDrawSegment(hps, 1L);
```

GpiOutlinePath — Outline Path

```
#define INCL_GPIPATHS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiOutlinePath (HPS hps, LONG IPath, LONG IOptions)

This function draws the outline of a path.

Parameters

hps (HPS) — input

Presentation-space handle.

IPath (LONG) — input

Identifier of path to be outlined; it must be 1.

IOptions (LONG) — input

Options:

Reserved; must be 0.

Returns

Correlation and error indicators:

GPI_OK Successful

GPI_HITS Correlate hits

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PATH_ID

An invalid path identifier parameter was specified.

PMERR_INV_RESERVED_FIELD

An invalid reserved field was specified.

PMERR_PATH_UNKNOWN

An attempt was made to perform a path function on a path that did not exist.

Remarks

The outline of the path is drawn, using the line attributes, including cosmetic line width (see GpiSetLineWidth) but not geometric line width (see GpiSetLineWidthGeom). This normally has the same effect as if the lines, curves, and so on, which comprise the path, had been drawn without defining them as being within a path. However, if character strings (referencing outline fonts) are contained within the path, the outlines of the characters, without the interior fill, are drawn by GpiOutlinePath, giving the appearance of hollow characters.

Open figures within the path are not closed automatically.

When the outline of the path has been drawn, the path is deleted.

Related Functions

- GpiBeginPath
- GpiEndPath
- GpiFillPath
- GpiModifyPath
- GpiPathToRegion
- GpiSetClipPath
- GpiStrokePath
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

Graphic Elements and Orders

Element Type: OCODE_GOPTH

Order: Outline Path

Example Code

This example uses GpiOutlinePath to draw the outline of a path (in this case a triangle).

```
#define INCL_GPIPATHS          /* Path functions          */
#include <os2.h>

LONG  lHits;                  /* correlation/error indicator */
HPS    hps;                   /* Presentation-space handle    */
POINTL ptlStart = { 0, 0 }; /* first vertex                  */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices */

GpiBeginPath(hps, 1L);        /* start the path bracket */
GpiMove(hps, &ptlStart);      /* move to starting point */
GpiPolyLine(hps, 2L, ptlTriangle); /* draw the three sides */
GpiCloseFigure(hps);          /* close the triangle      */
GpiEndPath(hps);              /* end the path bracket    */

.
.
lHits = GpiOutlinePath(hps, 1L, 0L);
```

GpiPaintRegion — Paint Region

```
#define INCL_GPIREGIONS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiPaintRegion (HPS hps, HRGN hrgn)

This function paints a region into a presentation space, using the current pattern attributes.

Parameters

hps (HPS) — input
Presentation-space handle.

hrgn (HRGN) — input
Region handle.

Returns

Correlation and error indicators:

GPI_OK Successful

GPI_HITS Correlate hits

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_REGION_IS_CLIP_REGION	An attempt was made to perform a region operation on a region that is selected as a clip region.
PMERR_INV_HRGN	An invalid region handle was specified.
PMERR_HRGN_BUSY	An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

Remarks

The current GPI area foreground and background colors are used. Mixing is controlled by the area foreground mix only.

It is invalid if the specified region is currently selected as the clip region (by GpiSetClipRegion).

The region is assumed to be defined in device coordinates.

Note: This function must not be used when creating SAA-conforming metafiles; see "Metafile Restrictions" on page G-1.

Related Functions

- GpiBeginArea
- GpiBeginPath
- GpiFillPath
- WinFillRect
- GpiCombineRegion
- GpiCreateRegion
- GpiDestroyRegion
- GpiEqualRegion
- GpiOffsetRegion
- GpiPtInRegion
- GpiQueryRegionBox
- GpiQueryRegionRects
- GpiRectInRegion
- GpiSetRegion
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix
- GpiSetPattern
- GpiSetPatternRefPoint
- GpiSetPatternSet

Example Code

This example uses the GpiPaintRegion function to fill a complex region consisting of three, intersecting rectangles. The region is filled with a red, diagonal pattern.

```
#define INCL_GPIREGIONS      /* Region functions */
#include <os2.h>

HPS hps;          /* presentation space handle */
HRGN hrgn;        /* handle for region */
RECTL arcl[3] = { 100, 100, 200, 200, /* 1st rectangle */
                  150, 150, 250, 250, /* 2nd rectangle */
                  200, 200, 300, 300 }; /* 3rd rectangle */

hrgn = GpiCreateRegion(hps, 3L, arcl);
GpiSetColor(hps, CLR_RED);
GpiSetPattern(hps, PATSYM_DIAG1);
GpiPaintRegion(hps, hrgn);
```


GpiPartialArc – Partial Arc

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiPartialArc (HPS hps, PPOINTL pptlCenter, FIXED fxMultiplier, FIXED fxStartAngle, FIXED fxSweepAngle)

This function draws a straight line, followed by an arc.

Parameters

hps (HPS) – input
Presentation-space handle.

pptlCenter (PPOINTL) – input
Center point.
Center of the arc.

fxMultiplier (FIXED) – input
Multiplier.
This determines the size of the arc in relation to an arc with the current arc parameters.
The implementation limit for the multiplier is 255.
The value must *not* be negative.

fxStartAngle (FIXED) – input
Start angle in degrees.
The value must be positive.

fxSweepAngle (FIXED) – input
Sweep angle in degrees.
The value must be positive.

Returns

Correlation and error indicators:

GPI_OK	Successful
GPI_HITS	Correlate hits
GPI_ERROR	Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_MULTIPLIER	An invalid multiplier parameter was specified with GpiPartialArc or GpiFullArc.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.
PMERR_INV_ANGLE_PARM	An invalid angle parameter was specified with GpiPartialArc.
PMERR_INV_NESTED_FIGURES	Nested figures have been detected within a path definition.

Remarks

This function draws two figures:

- A straight line, from the current position to the starting point of an arc
- An arc, with its center at the specified point.

The full arc, of which the arc is a part, is identical to that defined by GpiFullArc. The part of the arc drawn by this primitive is defined by the parameters *fxStartAngle* and *fxSweepAngle*, that are the start and sweep angles, subtended from the center, if the current arc parameters specify a circular form. If they do not, these angles are skewed to the same degree that the ellipse is a skewed circle. *fxStartAngle* is measured counterclockwise from the x axis of the circle before application of the arc parameters. Both angles must be positive; whether the arc is drawn clockwise or counterclockwise is determined by the arc parameters.

Current position is updated to the final point on the arc.

Note: This differs from GpiFullArc, where current position remains at the center of the figure. A primitive (such as GpiLine) following GpiPartialArc draws from the end point of the arc.

A segment of a pie can be drawn by the following calling sequence:

1. GpiMove, to center of pie
2. GpiPartialArc, drawing one spoke and the arc
3. GpiLine, back to center.

The third step can be performed implicitly by autoclosure if an area is being drawn.

A closed figure bounded by a chord and an arc can be drawn by the following calling sequence:

1. GpiSetLineType to invisible
2. GpiPartialArc, with *fxStartAngle* = angle2, and *fxSweepAngle* = 0, to define one end of the chord
3. GpiSetLineType to visible
4. GpiPartialArc, with *fxStartAngle* = angle1, and *fxSweepAngle* = angle2 – angle1.

(In the second example, angle2 is greater than angle1. If the interior of the chord is to be shaded, the area must start after step 2 or 3.)

A sweep angle of greater than 360 degrees is valid, and means that after the initial line a full arc is drawn, followed by a partial arc with a sweep angle of (*fxSweepAngle* MOD 360) degrees.

Related Functions

- GpiFullArc
- GpiPointArc
- GpiSetArcParams
- GpiSetDefArcParams
- GpiSetLineType
- GpiSetLineWidth
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

GpiPartialArc — Partial Arc

Graphic Elements and Orders

Element Type: OCODE_GCPARC

Order: Partial Arc at Current Position

Example Code

This example uses the GpiPartialArc function to draw a chord (an arc whose end points are connected by a straight line).

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                        /* presentation space handle */
POINTL ptl = { 100, 100 };      /* center point for arc */

GpiSetLineType(hps, LINETYPE_INVISIBLE);
GpiPartialArc(hps, &ptl, MAKEFIXED(50, 0), MAKEFIXED(0, 0),
              MAKEFIXED(180, 0));
GpiSetLineType(hps, LINETYPE_SOLID);
GpiPartialArc(hps, &ptl, MAKEFIXED(50, 0), MAKEFIXED(0, 0),
              MAKEFIXED(180, 0));
```

GpiPathToRegion – Path to Region

```
#define INCL_GPIPATHS /* Or use INCL_GPI or INCL_PM */
```

HRGN GpiPathToRegion (HPS hps, LONG lPath, ULONG flOptions)

This function converts a path to a region.

Parameters

hps (HPS) – input

Presentation-space handle.

lPath (LONG) – input

Identifier of path to be converted; it must be 1.

flOptions (ULONG) – input

Fill options:

FPATH_ALTERNATE Fills the path using the alternate rule; see GpiBeginArea.

FPATH_WINDING Fills the path using the winding rule; see GpiBeginArea. This value must be selected if the path has been modified using GpiModifyPath.

The default is FPATH_ALTERNATE.

Returns

Region handle:

≠0 Region handle

RGN_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PATH_ID

An invalid path identifier parameter was specified.

PMERR_INV_PATH_CONVERT_OPTIONS

An invalid options parameter was specified with GpiOutlinePath.

PMERR_PATH_UNKNOWN

An attempt was made to perform a path function on a path that did not exist.

Remarks

This function converts a path (originally defined by a series of GPI drawing calls) to a region. The new region can be operated on by the GPI region calls; in particular GpiCombineRegion can be used to combine it with another region.

Any open figures within the path are closed automatically.

The boundaries of the area defined by the path are considered to be part of the interior, so that a point on the boundary is included in the new region.

After a path is converted to a region, it no longer exists as a path. The path cannot be reused for any other purpose.

GpiPathToRegion —

Path to Region

Related Functions

- GpiBeginPath
- GpiCombineRegion
- GpiEndPath
- GpiFillPath
- GpiModifyPath
- GpiOutlinePath
- GpiSetClipPath
- GpiStrokePath

Example Code

This example uses GpiPathToRegion to convert a path (a triangle) to a region using the winding rule to fill the region.

```
#define INCL_GPIPATHS          /* Path functions          */
#include <os2.h>

HRGN  hrgn;          /* handle for region          */
HPS   hps;           /* Presentation-space handle  */
POINTL ptlStart = { 0, 0 }; /* first vertex              */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices */

GpiBeginPath(hps, 1L);          /* start the path bracket */
GpiMove(hps, &ptlStart);        /* move to starting point */
GpiPolyLine(hps, 2L, ptlTriangle); /* draw the three sides */
GpiCloseFigure(hps);            /* close the triangle */
GpiEndPath(hps);                /* end the path bracket */

hrgn = GpiPathToRegion(hps, 1L, FPATH_WINDING);
```

GpiPlayMetaFile – Play Metafile

```
#define INCL_GPIMETAFILES /* Or use INCL_GPI or INCL_PM */
```

```
LONG GpiPlayMetaFile (HPS hps, HMF hmf, LONG ICount1, PLONG alOptarray,  
PLONG piSegCount, LONG ICount2, PSZ pszDesc)
```

This function plays a metafile into a presentation space.

Parameters

hps (HPS) – input
Presentation-space handle.

hmf (HMF) – input
Metafile handle.

Handle of the metafile containing the data.

ICount1 (LONG) – input
Count of elements in *alOptarray*.

alOptarray (PLONG) – input
Array of options for playing.

The values of the elements in this array determine what action is to be taken when the metafile is played into the specified presentation space. The elements in the array are numbered consecutively, starting with `PMF_SEGBASE`. The element number constants start with 0. (Refer to the appropriate bindings reference.) Any elements in the array that are not set to one of the values defined below must be set to 0.

Optarray.[PMF_SEGBASE]

Reserved; must be 0.

Optarray.[PMF_LOADTYPE]

Specifies what transformations should be performed on the imported picture. The options are:

LT_DEFAULT The default; same as `LT_NOMODIFY`

LT_NOMODIFY The graphics are restored using the current viewing transform (see `GpiSetViewingTransformMatrix`), rather than the ones that were in use when the data was created. This is the default action.

Any change to the graphics field or default viewing transform during the course of the picture will be ignored if this option is specified (or defaulted).

LT_ORIGINALVIEW The graphics are restored using the viewing transforms that are in the metafile.

The default viewing transform of the presentation space is not altered (unless `RES_RESET` is specified). However, any changes to the default viewing transform that occur during the course of the picture (and also any graphics field clipping) cause changes to the values in the presentation space.

Optarray.[PMF_RESOLVE]

Reserved; must be 0.

Optarray.[PMF_LCIDS]

Specifies the action to be taken for any logical font definitions, or bit maps referenced by local identifiers for use as shading patterns that are held in the metafile.

The options are:

GpiPlayMetaFile —

Play Metafile

Optarray.[PMF_RESET]

LC_DEFAULT Default; same as LC_NOLOAD.

LC_NOLOAD Do not load such objects. This is the default, and is used where the application expects the correct objects to be already loaded.

LC_LOADDISC Load all objects referenced in the metafile, first deleting any already existing in the presentation space, for which the referenced local identifier is already in use.

Specifies whether the presentation space should be reset before playing the metafile, with the page units and size being set as defined in the metafile.

The options are:

RES_DEFAULT Default; same as RES_NORESET.

RES_NORESET Do not perform a reset.

RES_RESET Reset the presentation space, before loading any logical fonts, color tables, segments, and so on, as follows:

1. Reset the page units and page size to the values contained in the metafile.
2. Set up default transformations, based on the page units and size, as if the presentation space had just been created with these values.
3. Further modify the device transform to ensure that the physical size of the metafile picture is preserved. (Only performed if the page units in the metafile are not PU_ARBITRARY or PU_PELS.)
4. Perform the equivalent of GpiResetPS (option GRES_ALL).
5. Set the default viewing transform to the value specified in the metafile.

This option should normally be used with a PMF_LOADTYPE option of LT_ORIGINALVIEW and LC_LOADDISC, but this is not enforced.

Optarray.[PMF_SUPPRESS]

Specifies whether the playing of this metafile actually occurs. This is provided to allow an application to use the PMF_RESET option, and then to regain control to perform further presentation-space modifications if necessary, before playing the remainder of the metafile.

The options are:

SUP_DEFAULT Default; same as SUP_NOSUPPRESS.

SUP_NOSUPPRESS Do not suppress the remainder of the metafile.

SUP_SUPPRESS Suppress the remainder of the metafile.

If this option is selected, only processing as determined by the PMF_RESET option is performed. The remainder of the metafile, and all other options, are ignored.

GpiPlayMetaFile – Play Metafile

Optarray.[PMF_COLORTABLES]

Specifies the action to be taken with respect to any color table or palette implied or present within the metafile.

The options are:

CTAB_DEFAULT Default; same as CTAB_NOMODIFY.

CTAB_NOMODIFY Ignore. The default or loaded color table or selected palette in the presentation space is unchanged, as are the references to color attributes in the new data. This is the default; it is suitable where it is known that the currently loaded color table or selected palette (if any) is suitable for the use of color in the imported picture.

CTAB_REPLACE Overwrite the currently-loaded color table (if any), with a color table as implied or present in the metafile. This can be used where there is no existing picture.

CTAB_REPLACEPALETTE Overwrite the currently-selected palette (if any), with a palette as implied or present in the metafile. This can be used where there is no existing picture.

Note: If the metafile specifies a color table in RGB mode, the currently-selected palette (if any) is overwritten with a color table in RGB mode, and a warning is issued.

Optarray.[PMF_COLORREALIZABLE]

Specifies whether the color table data contained in the metafile should be loaded with the LCOL_REALIZABLE option or not (see GpiCreateLogColorTable).

The options are:

CREA_DEFAULT Default; same as CREA_NOREALIZE

CREA_DOREALIZE Load the color table with the realizable option set, and realize the color table.

CREA_NOREALIZE Load the color table with the realizable option off. This is the default.

Optarray.[PMF_DEFAULTS]

Specifies how the drawing defaults contained in the metafile should be used (see GpiSetDefAttrs, GpiSetDefViewingLimits, GpiSetDefTag, and GpiSetDefArcParams).

The options are:

DDEF_DEFAULT Default; same as DDEF_IGNORE

DDEF_IGNORE Ignore any drawing default values in the metafile.

DDEF_LOADDISC Change any drawing default values in the presentation space that are specified in the metafile, to the values contained in the metafile.

plSegCount (PLONG) – output
Reserved.

The value 0 is always returned.

lCount2 (LONG) – input
Count of bytes in *pszDesc*.

GpiPlayMetaFile — Play Metafile

pszDesc (PSZ) — output
Descriptive record.

pszDesc is a buffer that, on return, contains the descriptive record, of up to 253 bytes, that is saved when the metafile is created (see *DevOpenDC*). This is null-terminated, even if it has to be truncated.

Returns

Correlation and error indicators:

GPI_OK Successful
GPI_HITS Correlate hits
GPI_ERROR Error.

Possible returns from *WinGetLastError*

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_HMF	An invalid metafile handle was specified.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_INV_PLAY_METAFILE_OPTION	An invalid option parameter was specified with <i>GpiPlayMetaFile</i> .
PMERR_INCOMPATIBLE_METAFILE	An attempt was made to associate a presentation space and a metafile device context with incompatible page units, size or coordinate format; or to play a metafile using the <i>RES_RESET</i> option (to reset the presentation space) to a presentation space that is itself associated with a metafile device context.
PMERR_INV_METAFILE	An invalid metafile was specified with <i>GpiPlayMetaFile</i> .
PMERR_INV_MICROPS_ORDER	An attempt was made to play a metafile containing orders that are invalid in a micro presentation space.
PMERR_STOP_DRAW_OCCURRED	Segment drawing or <i>GpiPlayMetaFile</i> was stopped prematurely in response to a <i>GpiSetStopDraw</i> request.
PMERR_INV_OUTSIDE_DRAW_MODE	An attempt was made to issue a <i>GpiSavePS</i> or <i>GpiRestorePS</i> function, or an output only function (for example, <i>GpiPaintRegion</i>) from <i>GpiPlayMetaFile</i> without the drawing mode set to <i>DM_DRAW</i> .
PMERR_INV_ELEMENT_POINTER	An attempt was made to issue <i>GpiPutData</i> with the element pointer not pointing at the last element.
PMERR_INV_IN_CURRENT_EDIT_MODE	An attempt was made to issue a function invalid inside the current editing mode.
PMERR_PROLOG_ERROR	A prolog error was detected during drawing. Segment prologs are used internally within retained segments and also appear in metafiles. This error can also arise from an End Prolog order that is outside a prolog.
PMERR_DUP_SEG	During <i>GpiPlayMetaFile</i> , while the actual drawing mode was <i>draw-and-retain</i> or <i>retain</i> , a metafile segment to be stored in the presentation space was found to have the same segment identifier as an existing segment.

GpiPlayMetaFile – Play Metafile

Remarks

This function executes the contents of a metafile. This process is known as “playing” the metafile. Whether the graphics are drawn, or retained in segment store, or both, depends upon the current drawing mode (see GpiSetDrawingMode) in the presentation space, for the chained and unchained segment contexts, as appropriate. If chained segments are retained, they are added to the end of any existing segment chain. An error is raised if a segment is to be retained, and it has the same (nonzero) identifier as a currently existing segment.

A segment must not be open when this function is issued. At the completion of the call, there is no open segment.

The application may need to reset the presentation space by GpiResetPS, before issuing this function. Alternatively, the PMF_RESET option on this function may be suitable.

Segments retain the segment attributes that they originally possessed.

Related Functions

- GpiCopyMetaFile
- GpiDeleteMetaFile
- GpiLoadMetaFile
- GpiQueryMetaFileBits
- GpiQueryMetaFileLength
- GpiSaveMetaFile
- GpiSetMetaFileBits

GpiPlayMetaFile —

Play Metafile

Example Code

This example uses the GpiPlayMetaFile function to play (execute) the metafile loaded by GpiLoadMetaFile into a presentation space associated with a window. GpiPlayMetaFile is called twice: the first call resets the presentation space (by combining the RES_RESET and SUP_SUPPRESS flags), while the second call actually executes the metafile.

```
#define INCL_GPIMETAFILES      /* Metafile functions      */
#define INCL_GPICONTROL      /* GPI control Functions */
#include <os2.h>

HAB    hab;          /* anchor-block handle      */
HPS    hps;          /* presentation space handle */
HMF    hmf;          /* metafile handle          */
HDC    hdc;          /* Device-context handle     */
HWND   hwnd;         /* window handle            */
SIZEL  sizl={0,0};   /* use same page size as device */
CHAR   szBuffer[80]; /* descriptive record buffer */
LONG   lHits;        /* correlation/error indicator */

/* play metafile options array */
LONG optArray[PMF_DEFAULTS+1] =
    {0,LT_DEFAULT,0,LC_DEFAULT,RES_RESET,
     SUP_SUPPRESS,CTAB_DEFAULT,CREA_DEFAULT,
     DDEF_DEFAULT};

hmf = GpiLoadMetaFile(hab, "sample.met");

/* create window device context and presentation space,
   associating DC with the PS */
hdc = WinOpenWindowDC(hwnd);
hps = GpiCreatePS(hab, hdc, &sizl, PU_PELS | GPIA_ASSOC);

/* reset presentation space */
lHits = GpiPlayMetaFile(hps, hmf, 9L, optArray, (LONG *)0, 80L,
    szBuffer);

/* display metafile in window (reset and
   suppress flags must be changed) */
optArray[PMF_SUPPRESS]=SUP_DEFAULT;
optArray[PMF_RESET]=RES_DEFAULT;
lHits = GpiPlayMetaFile(hps, hmf, 9L, optArray, (LONG *)0, 80L,
    szBuffer);
```

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiPointArc (HPS hps, PPOINTL aptlPoints)

This function creates an arc, using the current arc parameters, through three points, starting at the current position.

Parameters

hps (HPS) – input
Presentation-space handle.

aptlPoints (PPOINTL) – input
Intermediate and end points.

Returns

Correlation and error indicators:

GPI_OK Successful

GPI_HITS Correlate hits

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.
PMERR_INV_NESTED_FIGURES	Nested figures have been detected within a path definition.

Remarks

The first element of the *aptlPoints* array defines an intermediate point along the arc, and the second element identifies the end point of the arc. Upon completion, current position is set to the end point of the arc.

Related Functions

- GpiFullArc
- GpiPartialArc
- GpiSetArcParams
- GpiSetDefArcParams
- GpiSetLineType
- GpiSetLineWidth
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

GpiPointArc —

Point Arc

Graphic Elements and Orders

Element Type: OCODE_GCARC

Order: Arc at Current Position

Example Code

This example uses the GpiPointArc function to draw an arc through the three points of a triangle. The GpiPolyLine function then draws the triangle.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                        /* presentation space handle */
POINTL ptlTriangle[] = { 0, 0, 100, 100, 200, 0 };

GpiMove(hps, &ptlTriangle[0]);  /* moves to start point (0, 0)*/
GpiPointArc(hps, &ptlTriangle[1]); /* draws the arc              */
GpiMove(hps, &ptlTriangle[0]);  /* moves to start point (0, 0)*/
/* draws the triangle */
GpiPolyLine(hps, 3L, &ptlTriangle[1]);
```

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiPolyFillet (HPS hps, LONG ICount, PPOINTL aptlPoints)

This function draws a curve starting at the current position and defined by the points supplied.

Parameters

hps (HPS) – input
Presentation-space handle.

ICount (LONG) – input
Number of points.
Must not be negative. Zero is valid but causes no output.

aptlPoints (PPOINTL) – input
Array of points.

Returns

Correlation and error indicators:

GPI_OK Successful

GPI_HITS Correlate hits

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.
PMERR_INV_NESTED_FIGURES	Nested figures have been detected within a path definition.

Remarks

If two points are supplied, an imaginary straight line is drawn from the current position to the first point and a second straight line from the first point to the second. A curve is then constructed, starting at the current position and tangential to the first straight line. The curve is drawn such that it reaches the last point at a tangent to the second straight line. Figure 5-1 on page 5-202 shows the curve constructed, given current position A and the two points B and C.

If more than two points are supplied, a series of imaginary straight lines is constructed through them (as in the GpiPolyLine function). All of the straight lines except the first and last are then divided in two at their mid-points. A series of curved fillets is then drawn, each starting at the end point of the last, at one of the mid-points. Figure 5-2 on page 5-202 shows the curve constructed, given current position A and three points B, C, and D.

The current position is set to the last point.

Each individual fillet always lies within the area bounded by the start, end, and control points.

It is not an error for any of the points to be coincident.

GpiPolyFillet — Polyfillet

The maximum number of fillets allowed in the polyfillet is more than 4 000.

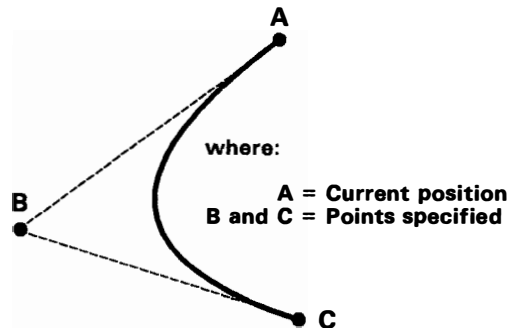


Figure 5-1. GpiPolyFillet Example A

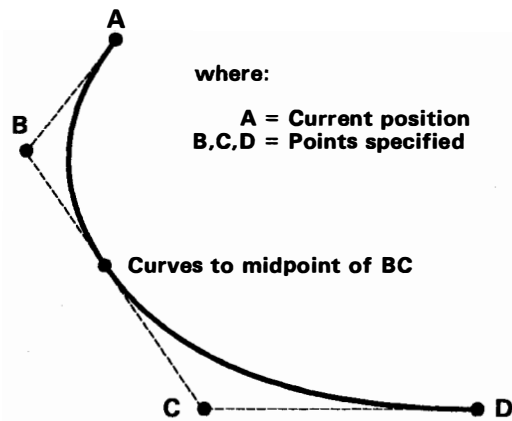


Figure 5-2. GpiPolyFillet Example B

Related Functions

- GpiPointArc
- GpiPolyFilletSharp
- GpiPolySpline
- GpiSetArcParams
- GpiSetDefArcParams
- GpiSetLineType
- GpiSetLineWidth
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

Graphic Elements and Orders

Element Type: OCODE_GCFLT

Order: Fillet at Current Position

As many of these orders are generated as is necessary to hold the specified fillets.

Example Code

This example uses the GpiPolyFillet function to draw a curve with a loop. The four points are the four points of a rectangle. The curve is drawn from the lower-left corner, through the midpoint of the top edge, and back to the lower-right corner.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions */
#include <os2.h>

HPS hps;                        /* presentation space handle */
POINTL ptlStart = { 0, 0 }; /* start point */
POINTL aptl[3] = { 200, 100, 0, 100, 200, 0 }; /* curve points */

GpiMove(hps, &ptlStart);        /* move to the lower-left corner */
GpiPolyFillet(hps, 3L, aptl);    /* draw the curve */
```


GpiPolyFilletSharp – Polyfillet Sharp

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiPolyFilletSharp (HPS hps, LONG ICount, PPOINTL aptlPoints, PFIXED afxSharpness)

This function creates a fillet on a series of connected lines, with the first line starting at the current position. Subsequent points identify the end points of the lines.

Parameters

hps (HPS) – input
Presentation-space handle.

ICount (LONG) – input
Count of points.

This is the number of points specified in *aptlPoints*. It must be $2*f$, where f is the number of fillets; the value must be a positive even number. Zero is valid but causes no output.

aptlPoints (PPOINTL) – input
An array of points.

These points are set as follows:

$c_1, e_1, c_2, e_2, c_3, e_3, \dots, c_f, e_f$

where:

c_f is the control point for the f 'th fillet

e_f is the end point of the f 'th fillet.

afxSharpness (PFIXED) – input
Array of sharpness values.

These give the sharpness of successive fillets.

Returns

Correlation and error indicators:

GPI_OK Successful

GPI_HITS Correlate hits

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT An invalid length or count parameter was specified.

PMERR_INV_COORDINATE An invalid coordinate value was specified.

PMERR_INV_SHARPNESS_PARM An invalid sharpness parameter was specified with GpiPolyFilletSharp.

PMERR_INV_NESTED_FIGURES Nested figures have been detected within a path definition.

GpiPolyFilletSharp – Polyfillet Sharp

Remarks

The first fillet is drawn using the two imaginary lines, one from current position to its control point (the first point specified in *aptlPoints*), and one from this point to the second point specified in *aptlPoints*. The fillet starts from current position, and ends at this second point. It is tangential to the first line at current position, and to the second line at the second point of *aptlPoints*. The sharpness of this fillet is given by the first element of the *afxSharpness* array.

Each subsequent fillet is drawn starting from the end point of the previous fillet, and uses the next two lines in the sequence, in a similar way. Therefore two points and one sharpness value are required for each fillet.

The differences from GpiPolyFillet are:

- The sharpness of each fillet is explicitly specified.
- Both the control and the end point of each fillet are explicitly specified.
- Adjacent fillets, generally, have a discontinuity in gradient, unless the points are chosen so that this is not the case.

The sharpness of each fillet is defined as follows. Let A and C be the start and end points, respectively, of the fillet, and let B be the control point. (See Figure 5-3.) Let W be the mid-point of AC. Let D be the point where the fillet intersects WB.

sharpness = WD/DB

so that

> 1.0 means a hyperbola is drawn

= 1.0 means a parabola is drawn

< 1.0 means an ellipse is drawn.

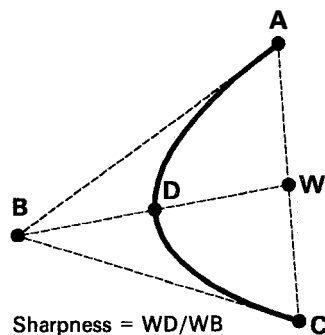


Figure 5-3. GpiPolyFilletSharp Example

On completion, the current position is the end point of the last line in the series. Each individual fillet always lies within the area bounded by the start, end, and control points.

It is not an error for any of the points to be coincident.

The maximum number of fillets allowed is more than 2 000.

GpiPolyFilletSharp — Polyfillet Sharp

Related Functions

- GpiPointArc
- GpiPolyFillet
- GpiPolySpline
- GpiSetArcParams
- GpiSetDefArcParams
- GpiSetLineType
- GpiSetLineWidth
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

Graphic Elements and Orders

Element Type: OCODE_GCSFLT

Order: Sharp Fillet at Current Position

As many of these orders are generated as is necessary to hold the specified fillets.

Example Code

This example uses the GpiPolyFilletSharp function to draw a curve with a loop. The curve is drawn within a rectangle. The sharpness values are chosen to draw the curve close to the control points.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions */
#include <os2.h>

HPS hps;                        /* presentation space handle */
POINTL ptlStart = { 0, 0 };    /* start of curve */
POINTL aptl[4]={ 100, 100, 200, 100, 0, 100, 200, 0}; /* points */
FIXED afx[2]={MAKEFIXED(4, 0), MAKEFIXED(4, 0)}; /* sharpness */

GpiMove(hps, &ptlStart);        /* move to first starting point */
GpiPolyFilletSharp(hps,         /* presentation-space handle */
    4L,                         /* 4 points in the array */
    aptl,                       /* address of array of points */
    afx);                       /* address of array of sharpness values */
```

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiPolygons (HPS hps, LONG ICount, PPOLYGON alPolygons, LONG IOptions, LONG lmodel)

This function draws a set of closed polygons.

Parameters

hps (HPS) – input
Presentation-space handle.

ICount (LONG) – input
Number of polygons.

Equal to the number of polygons in the polygons array. May be zero or positive, zero causes no output.

alPolygons (PPOLYGON) – input
Array of polygons.
An array of POLYGON structures.

IOptions (LONG) – input
Drawing options.

This contains fields of option bits. For each field, one value should be selected (unless the default is suitable). These values can be ORed together to determine whether to draw boundary lines as well as the area interior:

POLYGON_NOBOUNDARY Do not draw boundary lines

POLYGON_BOUNDARY Draw boundary lines (the default).

Construction of the area interior:

POLYGON_ALTERNATE Construct interior in *alternate* mode (the default)

POLYGON_WINDING Construct interior in *winding* mode.

lmodel (LONG) – input
Drawing model.

POLYGON_INCL The fill is inclusive of bottom right. This is the default.

POLYGON_EXCL The fill is exclusive of bottom right. This is provided to aid migration from other graphics models.

GpiPolygons – Draw Polygons

Returns

Correlation/error indicator:

GPI_OK	Successful
GPI_HITS	Correlate hits.
GPI_ERROR	Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_AREA_CONTROL	An invalid options parameter was specified with GpiBeginArea.
PMERR_INV_IN_PATH	An attempt was made to issue a function invalid inside a path bracket.
PMERR_ALREADY_IN_AREA	An attempt was made to begin a new area while an existing area bracket was already open.

Remarks

The polygons are filled using the current AREABUNDLE structure values. For the first polygon, the current position is the first point. For all subsequent polygons all points which define the polygon are given explicitly. The polygons are automatically closed, if necessary, by drawing a line from the last vertex to the first.

The polygons may overlap, but that is not necessary.

GpiPolygons is not valid inside of an area.

Graphic Elements and Orders

Element Type: **OCODE_GPOLYS**

Order: **Polygons**

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

LONG GpiPolyLine (HPS hps, LONG ICount, PPOINTL aptlPoints)

This function draws a series of straight lines starting at the current position and connecting the points specified.

Parameters

hps (HPS) – input
Presentation-space handle.

ICount (LONG) – input
Number of points
Must not be negative. Zero is valid but causes no output.

aptlPoints (PPOINTL) – input
Array of points.

Returns

Correlation and error indicators:

GPI_OK Successful

GPI_HITS Correlate hits

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.
PMERR_INV_NESTED_FIGURES	Nested figures have been detected within a path definition.

Remarks

On completion, current position is set to the last point.

The maximum number of lines allowed in a polyline is device dependent, but is always greater than 3 500 for GPIF_LONG format spaces and always greater than 7 200 for GPIF_SHORT format spaces (see the PS_FORMAT of GpiCreatePS for the meaning of this format).

GpiPolyLine — Polyline

Related Functions

- GpiBox
- GpiLine
- GpiPolyLineDisjoint
- GpiMove
- GpiSetCurrentPosition
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

Graphic Elements and Orders

Element Type: **OCODE_GCLINE**

Note that GpiLine also generates this element type.

Order: **Line at Current Position**

As many of these orders are generated as is necessary to hold the specified points.

Example Code

This example uses the GpiPolyLine function to draw a triangle.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions */
#include <os2.h>

HPS hps;                        /* presentation space handle */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices */

GpiMove(hps, &ptlTriangle[2]); /* moves to end point (0, 0)*/
GpiPolyLine(hps, 3L, &ptlTriangle[1]); /* draws triangle */
```

GpiPolyLineDisjoint – Polyline Disjoint

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

LONG GpiPolyLineDisjoint (HPS hps, LONG ICount, PPOINTL aptlPoints)

This function draws a series of disjoint straight lines using the end-point pairs specified.

Parameters

hps (HPS) – input
Presentation-space handle.

ICount (LONG) – input
Number of points

Must be even and not negative. Zero is valid, but it causes no output. The maximum number of points allowed is system-dependent, but it is at least 7 000.

aptlPoints (PPOINTL) – input
Array of points.

Returns

Correlation/error indicator:

GPI_OK Successful

GPI_HITS Correlate hit(s)

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT An invalid length or count parameter was specified.

PMERR_INV_COORDINATE An invalid coordinate value was specified.

PMERR_INV_NESTED_FIGURES Nested figures have been detected within a path definition.

Remarks

The effect of this function is the same as the following sequence of calls:

```
GpiMove (hps, Points[0]);
GpiLine (hps, Points[1]);
GpiMove (hps, Points[2]);
GpiLine (hps, Points[3]);
...
GpiMove (hps, Points[Count-2]);
GpiLine (hps, Points[Count-1]);
```

On completion, current position is set to the last point.

GpiPolyLineDisjoint – Polyline Disjoint

Related Functions

- GpiBox
- GpiLine
- GpiPolyLine
- GpiMove
- GpiSetCurrentPosition
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

Example Code

This example uses the GpiPolyLineDisjoint function to draw two lines.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                        /* presentation space handle */
POINTL ptlLines[] = { 100, 100, 100, 200, /* line 1 */
                      200, 100, 200, 200 }; /* line 2 */

GpiPolyLineDisjoint(hps, 4L, &ptlLines[1]); /* draw lines */
```

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiPolyMarker (HPS hps, LONG ICount, PPOINTL aptlPoints)

This function draws markers with their centers at each of a series of specified positions.

Parameters

hps (HPS) – input

Presentation-space handle.

ICount (LONG) – input

Number of points.

Must not be negative. Zero is valid but causes no output.

aptlPoints (PPOINTL) – input

Array of points.

A marker is drawn at each of these points.

Returns

Correlation and error indicators:

GPI_OK Successful

GPI_HITS Correlate hits

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

PMERR_INV_COORDINATE

An invalid coordinate value was specified.

Remarks

On completion, the current position is set to the position of the last marker in the series. The marker symbol is selected by the current values of the marker set and marker symbol attributes.

Related Functions

- GpiMarker
- GpiSetMarker
- GpiSetMarkerBox
- GpiSetMarkerSet
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

GpiPolyMarker — Polymarker

Graphic Elements and Orders

Element Type: **OCODE_GMRK**

Note that GpiMarker also generates this element type.

Order: **Marker at Given Position**

As many of these orders are generated as is necessary to hold the specified positions.

Example Code

This example uses the GpiPolyMarker function to draw a series of markers. It then uses the GpiPolyLine function to connect to markers with lines.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                        /* presentation space handle */
POINTL pt1Start = { 0, 0 }; /* start point                */
POINTL apt1[5]={10, 8, 20, 17, 30, 28, 40, 51, 50, 46};/* points*/

GpiPolyMarker(hps, 51, apt1);
GpiMove(hps, &pt1Start);
GpiPolyLine(hps, 5L, apt1);
```

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiPolySpline (HPS hps, LONG ICount, PPOINTL aptlPoints)

This function creates a succession of Bézier splines.

Parameters

hps (HPS) — input
Presentation-space handle.

ICount (LONG) — input
Count of points.

This is the number of points specified in *aptlPoints*. It must be three times the number of splines. The value must not be negative, and it must be divisible by 3. Zero is valid but causes no output.

aptlPoints (PPOINTL) — input
An array of points.

The points are given in this order:

c11, c12, e1, c21, c22, e2, ... cs1, cs2, es

where:

cs1 is the first control point of spline *s*
cs2 is the second control point of spline *s*
es is the end point of spline *s*.

Returns

Correlation and error indicators:

GPI_OK Successful
GPI_HITS Correlate hits
GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.
PMERR_INV_NESTED_FIGURES	Nested figures have been detected within a path definition.

Remarks

The first Bézier spline starts from the current position and goes to the third specified point, with the first and second points used as control points. Subsequent splines start from the ending point of the previous spline, and end at the next specified point but two, with the intervening points their first and second control points. It is the responsibility of the application to ensure that the gradient is continuous at each end and start point, if this is required.

GpiPolySpline — Polyspline

On completion, the current position is set to the last specified point. Each individual spline always lies within the area bounded by the start, end, and control points.

It is not an error for any of the points to be coincident.

The maximum number of splines allowed is more than 2 500.

Related Functions

- GpiPointArc
- GpiPolyFillet
- GpiPolyFilletSharp
- GpiSetArcParams
- GpiSetDefArcParams
- GpiSetLineType
- GpiSetLineWidth
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

Graphic Elements and Orders

Element Type: **OCODE_GCBEZ**

Order: **Bezier Spline at Current Position**

As many of these orders are generated as is necessary to hold the specified splines.

Example Code

This example uses the GpiPolySpline function to draw a curve. The curve is drawn within a skewed rectangle, with the bottom corners being the start and end points and the top corners being the control points.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions */
#include <os2.h>

HPS hps;                        /* presentation space handle */
POINTL ptlStart = { 0, 0 }; /* start point */
POINTL aptl[3] = { 0, 100, 200, 150, 200, 50 }; /* point array */

GpiMove(hps, &ptlStart);        /* moves to start point */
GpiPolySpline(hps,              /* presentation-space handle */
              3L,               /* 3 points in the array */
              aptl);            /* address of array of points */
```

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiPop (HPS hps, LONG ICount)

This function restores the primitive attributes that have been saved on the stack.

Parameters

hps (HPS) – input
Presentation-space handle.

ICount (LONG) – input
Number of attributes to be restored.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_LENGTH_OR_COUNT An invalid length or count parameter was specified.

PMERR_SEG_CALL_STACK_EMPTY A call stack empty condition was detected when attempting a pop function during GpiPop or segment drawing.

Remarks

Each time a primitive attribute call (such as color, or line type) is issued and the attribute mode is set to AM_PRESERVE, the values are put into a "Last in, First out" stack.

This function can reset the current attribute values (starting with the last one set) to the previous value; this is known as "popping." This allows a called segment to change the values of the attributes, and allows them to be restored on return to the caller (an implicit GpiPop function is performed for each preserved attribute when returning from a called segment).

When inside an area or path definition, this function is only valid if the attribute being popped is valid inside an area or path definition.

Note: It is not possible to check whether the attribute to be popped is valid before issuing this function.

GpiPop — Pop

Related Functions

- GpiQueryAttrMode
- GpiQueryAttrs
- GpiQueryDefAttrs
- GpiRestorePS
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiRestorePS

Graphic Elements and Orders

Element Type: **OCODE_GPOP**

Order: **Pop**

ICount of these orders are generated.

Example Code

This example uses the GpiPop function to restore the fill pattern and color attribute after painting a region.

```
#define INCL_GPIPRIMITIVES    /* GPI primitive functions    */
#define INCL_GPIREGIONS      /* GPI region functions  */
#include <os2.h>

HPS hps;          /* presentation space handle */
HRGN hrgn;        /* region handle             */

/* preserves attributes on stack */
GpiSetAttrMode(hps, AM_PRESERVE);
.
.
.
GpiSetColor(hps, CLR_RED);      /* sets color to red        */
GpiSetPattern(hps, PATSYM_DIAG1); /* sets pattern to a diagonal */
GpiPaintRegion(hps, hrgn);
GpiPop(hps, 2L); /* restores values of last two attributes set.*/
```

GpiPtInRegion – Point In Region

```
#define INCL_GPIREGIONS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiPtInRegion (HPS hps, HRGN hrgn, PPOINTL pptlPoint)

This function checks whether a point lies within a region.

Parameters

hps (HPS) – input

Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

hrgn (HRGN) – input

Region handle.

pptlPoint (PPOINTL) – input

Point to be checked.

The point is in device coordinates.

Returns

Inside and error indicators:

PRGN_OUTSIDE Not in region

PRGN_INSIDE In region

PRGN_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HRGN

An invalid region handle was specified.

PMERR_INV_COORDINATE

An invalid coordinate value was specified.

PMERR_REGION_IS_CLIP_REGION

An attempt was made to perform a region operation on a region that is selected as a clip region.

PMERR_HRGN_BUSY

An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

Remarks

It is invalid if the specified region is currently selected as the clip region (by GpiSetClipRegion).

GpiPtInRegion — Point In Region

Related Functions

- GpiCombineRegion
- GpiCreateRegion
- GpiDestroyRegion
- GpiEqualRegion
- GpiOffsetRegion
- GpiPaintRegion
- GpiQueryRegionBox
- GpiQueryRegionRects
- GpiRectInRegion
- GpiSetRegion

Example Code

This example uses GpiPtInRegion to determine if the point (150,150) lies within a region.

```
#define INCL_GPIREGIONS      /* Region functions          */
#include <os2.h>

LONG  lInside;              /* inside/error indicator      */
HPS    hps;                 /* Presentation-space handle   */
HRGN   hrgn;                /* handle for region           */
POINTL pptlPoint = {150L,150L}; /* point to be checked        */
RECTL  arcl[3] = { 100, 100, 200, 200, /* 1st rectangle              */
                  150, 150, 250, 250, /* 2nd rectangle              */
                  200, 200, 300, 300 }; /* 3rd rectangle              */

/* create a region comprising three rectangles */
hrgn = GpiCreateRegion(hps, 3L, arcl);

lInside = GpiPtInRegion(hps, hrgn, &pptlPoint);
```

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiPtVisible (HPS hps, PPOINTL pptlPoint)

This function checks whether a point is visible within the clipping region of the device associated with the specified presentation space.

Parameters

hps (HPS) – input
Presentation-space handle.

pptlPoint (PPOINTL) – input
Point to be checked.
The point is given in world coordinates.

Returns

Visibility indicator:

PVIS_INVISIBLE Not visible
PVIS_VISIBLE Visible
PVIS_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.

Remarks

For the purposes of this function, the clipping region is defined as the intersection between the application clipping region, and any other clipping, including windowing.

Related Functions

- GpiExcludeClipRectangle
- GpiIntersectClipRectangle
- GpiOffsetClipRegion
- GpiQueryClipBox
- GpiQueryClipRegion
- GpiQueryPel
- GpiRectVisible
- GpiSetClipRegion
- GpiSetGraphicsField
- WinExcludeUpdateRegion

GpiPtVisible — Point Visible

Example Code

This example uses GpiPtVisible to check whether (150,150) is visible within the clipping region of the device associated with the presentation space.

```
#define INCL_GPIPRIMITIVES    /* Primitive functions      */
#include <os2.h>

LONG  lVisibility;    /* visibility indicator      */
HPS   hps;            /* Presentation-space handle */
POINTL pptlPoint = {150L,150L}; /* point to be checked      */

lVisibility = GpiPtVisible(hps, &pptlPoint);
```

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiPutData (HPS hps, LONG IFormat, PLONG piLength, PBYTE pbData)

This function passes a buffer of graphics orders to the current segment, or draws the orders, or both of these. For details of the orders, see Chapter 33, "Graphics Orders."

Parameters

hps (HPS) — input
Presentation-space handle.

IFormat (LONG) — input
Coordinate type used:

DFORM_NOCONV	No coordinate conversion performed
DFORM_S370SHORT	S/370 format short (2-byte) integers
DFORM_PCSHORT	PC format short (2-byte) integers
DFORM_PCLONG	PC format long (4-byte) integers.

piLength (PLONG) — input/output
Length of graphic data.

Set by the application to the length of order data in *pbData*. If an incomplete order occurred, it is updated, on return, to the offset of the start of the incomplete order.

piLength must not be greater than 63 KB.

pbData (PBYTE) — input
Orders to be copied.

Returns

Correlation and error indicators:

GPI_OK	Successful
GPI_HITS	Correlate hits
GPI_ERROR	Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_PUTDATA_FORMAT	An invalid format parameter was specified with GpiPutData.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.
PMERR_DATA_TOO_LONG	An attempt was made to transfer more than the maximum permitted amount of data (64512 bytes) using GpiPutData, GpiGetData, or GpiElement.

GpiPutData — Put Data

PMERR_INV_ELEMENT_POINTER	An attempt was made to issue GpiPutData with the element pointer not pointing at the last element.
PMERR_INV_REPLACE_MODE_FUNC	An attempt was made to issue GpiPutData with the editing mode set to SEGEM_REPLACE.
PMERR_ORDER_TOO_BIG	An internal size limit was exceeded while converting orders from short to long format during GpiPutData processing. An order was too long to convert.

Remarks

The orders passed may be added to the current segment, drawn immediately, or both, depending on the current drawing mode (see GpiSetDrawingMode), and whether the primitives are within a segment.

If there is an incomplete order at the end of the buffer, *pLength* is updated to point to the start of the incomplete order. The application can then concatenate this partial order in front of the next buffer.

The orders End Prolog and Set Viewing Transform are not allowed.

This function is valid within an element bracket (see GpiBeginElement). It can contain GpiBeginElement and GpiEndElement orders, while these are in the correct sequence with respect to the currently opened segment in segment store.

The data in the buffer is converted, if necessary, to the presentation space format (defined when the presentation space is first created; see GpiCreatePS).

This function is invalid if the editing mode (see GpiSetEditMode) is set to SEGEM_REPLACE, and also in SEGEM_INSERT mode if the element pointer is not pointing to the last element.

Related Functions

- GpiBeginElement
- GpiEndElement
- GpiGetData

Example Code

This example uses the GpiPutData function to copy graphics orders from one segment to another.

```
#define INCL_GPISEGMENTS      /* Segment functions      */
#include <os2.h>

HPS hps;                    /* presentation space handle */
LONG fFormat = DFORM_NOCONV; /* do not convert coordinates */
LONG offSegment = 0L;       /* offset in segment */
LONG offNextElement = 0;    /* offset in segment to next element */
LONG cb = 0L;               /* bytes retrieved */
BYTE abBuffer[512];         /* data buffer */

GpiOpenSegment(hps, 3L);    /* open segment to receive the data */
do {
    offSegment += cb;
    offNextElement = offSegment;
    cb = GpiGetData(hps, 2L, &offNextElement, fFormat, 512L, abBuffer);

    /* Put data in other segment. */

    if (cb > 0L) GpiPutData(hps, /* presentation-space handle */
                             fFormat, /* format of coordinates */
                             &cb, /* number of bytes in buffer */
                             abBuffer); /* buffer with graphics-order data */
} while (cb > 0L);
GpiCloseSegment(hps);      /* close segment that received data */
```

GpiQueryArcParams — Query Arc Parameters

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryArcParams (HPS hps, PARCPARAMS parcpArcParams)

This function returns the current arc parameters used to draw full, partial, and 3-point arcs.

Parameters

hps (HPS) — input

Presentation-space handle.

parcpArcParams (PARCPARAMS) — output

Arc parameters.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from GetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE

An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE

An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

Arc parameters are set by GpiSetArcParams.

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Related Functions

- GpiQueryAttrs
- GpiSetArcParams

GpiQueryArcParams — Query Arc Parameters

Example Code

This example uses GpiQueryArcParams to return the current arc parameters used to draw full, partial, and 3-point arcs. The example queries the arc parameters and assigns a variable to the P coefficient if the query succeeds.

```
#define INCL_GPIPRIMITIVES    /* Primitive functions      */
#include <os2.h>

BOOL fSuccess;               /* success indicator      */
HPS hps;                     /* Presentation-space handle */
ARCPARAMS parcpArcParams; /* Arc parameters          */
LONG lPcoefficient;          /* p coefficient of arc definition */

fSuccess = GpiQueryArcParams(hps, &parcpArcParams);

/* if successful, assign value of P coefficient */
if (fSuccess == TRUE)
    lPcoefficient = parcpArcParams.lP;
```


GpiQueryAttrMode — Query Attribute Mode

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryAttrMode (HPS hps)

This function returns the current value of the attribute mode, as set by GpiSetAttrMode.

Parameters

hps (HPS) — input
Presentation-space handle.

Returns

Current attribute mode:
≥0 Current attribute mode
AM_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.

Related Functions

- GpiQueryAttrs
- GpiSetAttrMode

Example Code

This example uses GpiQueryAttrMode to return the current value of the attribute mode and sets a new mode using GpiSetAttrMode; after the application has finished using the new mode, the original attribute mode is restored.

```
#define INCL_GPIPRIMITIVES /* Primitive functions */
#include <os2.h>

LONG lMode; /* current attribute mode (or error) */
HPS hps; /* Presentation-space handle */

/* query current attribute mode */
lMode = GpiQueryAttrMode(hps);

/* set new mode */
GpiSetAttrMode(hps, AM_PRESERVE);
.
.
/* restore original mode */
GpiSetAttrMode(hps, lMode);
```

GpiQueryAttrs – Query Attributes

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryAttrs (HPS hps, LONG IPrimType, ULONG fIAttrMask, PBUNDLE ppbunAttrs)

This function returns current attributes for the specified primitive type.

Parameters

hps (HPS) – input
Presentation-space handle.

IPrimType (LONG) – input
Primitive type.

This is the type of primitive for which attributes are to be queried, as follows:

PRIM_LINE Line and arc primitives

PRIM_CHAR Character primitives

PRIM_MARKER Marker primitives

PRIM_AREA Area primitives

PRIM_IMAGE Image primitives.

fIAttrMask (ULONG) – input
Attributes mask.

Each flag that is set indicates that the corresponding flag in *IDefMask* is to be updated, and that if the corresponding attribute is not currently set to default, its value is to be returned in the *ppbunAttrs* buffer.

If all flags in *fIAttrMask* are zero, the *ppbunAttrs* buffer address is not used.

ppbunAttrs (PBUNDLE) – output
Attributes.

ppbunAttrs is a buffer in which is returned the value of each non-default attribute for which the *fIAttrMask* flag is set, in the order specified in *GpiSetAttrs* for the particular primitive type.

Only data for attributes for which the appropriate flag in *fIAttrMask* is set is updated, so *ppbunAttrs* need only be large enough for the highest offset attribute to be returned (see *GpiSetAttrs*).

The data returned in *ppbunAttrs* for any attribute for which the *fIAttrMask* flag is set, but which is currently set to default, is undefined.

Returns

Defaults mask.

As *fIDefMask* in *GpiSetAttrs*:

GPI_ALTEERROR Error occurred

Positive Defaults mask, numeric value can be greater than or equal to 0.

Possible returns from *WinGetLastError*

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

GpiQueryAttrs — Query Attributes

PMERR_INV_PRIMITIVE_TYPE	An invalid primitive type parameter was specified with GpiSetAttrs or GpiQueryAttrs.
PMERR_UNSUPPORTED_ATTR	An unsupported attribute was specified in the attrmask with GpiSetAttrs or GpiQueryAttrs.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**. This function returns a mask, similar in meaning to *flDefMask* in GpiSetAttrs. Each flag in the returned mask is updated if the corresponding flag in *flAttrMask* is set. It is set if the attribute is set to the default, otherwise it is reset. Other flags are undefined.

The parameters returned by this function may be used to reinstate exactly the same attributes as are queried, using GpiSetAttrs.

Related Functions

- GpiSetAttrs

Example Code

This example uses the GpiQueryAttrs function to retrieve the current attributes for the line primitive.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                        /* presentation space handle */
LINEBUNDLE lband;
LONG flDefMask;

flDefMask = GpiQueryAttrs(hps, /* presentation-space handle */
    PRIM_LINE,                /* line primitive            */
    LBB_COLOR |               /* line color                */
    LBB_MIX_MODE |           /* color-mix mode            */
    LBB_WIDTH |              /* line width                */
    LBB_GEOM_WIDTH |         /* geometric-line width      */
    LBB_TYPE |               /* line style                */
    LBB_END |                /* line-end style            */
    LBB_JOIN,                /* line-join style           */
    &lband);                  /* buffer for attributes     */

if (flDefMask & LBB_COLOR)
{
    /* The line color has the default value. */
}
```

GpiQueryBackColor – Query Background Color

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryBackColor (HPS hps)

This function returns the current value of the (character) background color attribute, as set by the GpiSetBackColor function.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Background color:

CLR_ERROR Error

CLR_DEFAULT Default

Otherwise Background color index.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE

An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE

An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Related Functions

- GpiQueryAttrs
- GpiSetBackColor

Example Code

This example uses GpiQueryBackColor to return the current value of the (character) background color attribute, as set by the GpiSetBackColor call.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions      */
#include <os2.h>

LONG  lColor;                  /* current background color (or error) */
HPS   hps;                    /* Presentation-space handle           */

lColor = GpiQueryBackColor(hps);
```

GpiQueryBackMix – Query Background Mix

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryBackMix (HPS hps)

This function returns the current value of the (character) background color-mixing mode, as set by the GpiSetBackMix function.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Background mix:

BM_DEFAULT Default
>0 Background mix mode
BM_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Related Functions

- GpiQueryAttrs
- GpiSetBackMix

Example Code

This example uses GpiQueryBackMix to return the current value of the (character) background color-mixing mode, as set by the GpiSetBackMix call.

```
#define INCL_GPIPRIMITIVES /* Primitive functions */
#include <os2.h>

LONG lMixMode; /* current background mix (or error) */
HPS hps; /* Presentation-space handle */

lMixMode = GpiQueryBackMix(hps);
```

GpiQueryBitmapBits – Query Bit-Map Bits

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryBitmapBits (HPS hps, LONG IScanStart, LONG IScans, PBYTE pbBuffer, PBITMAPINFO2 pbmi2InfoTable)

This function transfers data from a bit map to application storage.

Parameters

hps (HPS) – input

Presentation-space handle.

IScanStart (LONG) – input

Starting line number.

Scan-line number at which the data transfer is to start, counting from zero as the bottom line.

IScans (LONG) – input

Number of scan lines to be returned.

pbBuffer (PBYTE) – output

Data area.

Data area into which the bit-map data is copied.

pbmi2InfoTable (PBITMAPINFO2) – input/output

Bit-map information table.

Storage must be provided for the associated color table.

Returns

Number of scan lines actually returned:

≥0 Number of scan lines actually returned

GPI_ALTError Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

PMERR_INV_INFO_TABLE

An invalid bit-map info table was specified with a bit-map operation.

PMERR_NO_BITMAP_SELECTED

An attempt has been made to operate on a memory device context that has no bit map selected.

PMERR_INV_SCAN_START

An invalid scanstart parameter was specified with a bitmap function.

PMERR_INCORRECT_DC_TYPE

An attempt was made to perform a bit-map operation on a presentation space associated with a device context of a type that is unable to support bit-map operations.

GpiQueryBitmapBits — Query Bit-Map Bits

Remarks

The presentation space must be currently associated with a memory device context, which has a bit map currently selected.

The *pbmi2InfoTable* must be initialized by the application with the values of *cbFix*, and also *cPlanes* and *cBitCount*, set to the format required. The standard bit-map formats are supported, plus any known to be supported by the device (see *GpiQueryDeviceBitmapFormats*). Each of the following fields must also be set by the application before issuing the call (unless the *BITMAPINFO2* structure is truncated and the field is not present):

- *ulCompression*
- *usReserved*
- *usRecording*
- *usRendering*
- *ulColorEncoding*

This function returns the values of *cx*, *cy* (plus any other information, apart from that set by the application, for which space is available in the *BITMAPINFO2* structure), and the color table array filled in by the system.

The bit-map data is converted where necessary.

pbBuffer must point to a storage area large enough to contain data for the requested number of scan lines. The amount of storage required for one scan line can be determined by *GpiQueryBitmapParameters*. It is

$((\text{bitcount} * \text{bitmapwidth} + 31) / 32) * \text{planes} * 4$ bytes

The storage required for the entire bit map is this value multiplied by *bltmapheight*.

Related Functions

- *GpiSetBitmapBits*

GpiQueryBitmapBits – Query Bit-Map Bits

Example Code

This example uses GpiQueryBitmapBits to copy the image data of a bit map from a presentation space associated with a memory device context.

```
#define INCL_GPIBITMAPS          /* GPI Bit-map functions */
#define INCL_DOSMEMMGR          /* DOS Memory Manager Functions */
#include <os2.h>

HPS hps;                        /* presentation space handle */
BITMAPINFOHEADER2 bmp = { 16, 640, 350, 1, 1 }; /* info struct */
ULONG cbBuffer, cbBitmapInfo; /* buffer lengths */
PBYTE pbBuffer;                /* bit-map data buffer */
PBITMAPINFO2 pbmi;             /* info structure */

/*
 * Compute the size of the image-data buffer and the bit map
 * information structure.
 */
cbBuffer = (((bmp.cBitCount * bmp.cx) + 31) / 32)
           * 4 * bmp.cy * bmp.cPlanes;
cbBitmapInfo = sizeof(BITMAPINFO2) +
               (sizeof(RGB) * (1 << bmp.cBitCount));

/*
 * Allocate memory for the image data-buffer and the bit map
 * information structure.
 */
DosAllocMem((VOID *)pbBuffer, cbBuffer,
            PAG_COMMIT | PAG_READ | PAG_WRITE);
DosAllocMem((VOID *)pbmi, cbBitmapInfo,
            PAG_COMMIT | PAG_READ | PAG_WRITE);

/* Copy the image data. */
pbmi->cbFix = 16L;
pbmi->cPlanes = 1;
pbmi->cBitCount = 1;
GpiQueryBitmapBits(hps, 0L, (LONG) bmp.cy, pbBuffer, pbmi);
```


GpiQueryBitmapDimension — Query Bit-Map Dimension

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryBitmapDimension (HBITMAP hbm, PSIZEL psizlBitmapDimension)

This function returns the width and height of a bit map, as specified on a previous GpiSetBitmapDimension function.

Parameters

hbm (HBITMAP) — input
Bit-map handle.

psizlBitmapDimension (PSIZEL) — output
Size of bit map.

The width and height of the bit map in 0.1 millimeter units.

If not set by GpiSetBitmapDimension, zeros are returned.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HBITMAP

An invalid bit-map handle was specified.

PMERR_HBITMAP_BUSY

An internal bit map busy error was detected. The bit map was locked by one thread during an attempt to access it from another thread.

Related Functions

- GpiSetBitmapDimension

Example Code

This example uses GpiQueryBitmapDimension to return the width and height of a bit map, as specified on a previous GpiSetBitmapDimension call; if successful, it assigns the width to a variable.

```
#define INCL_GPIBITMAPS          /* Bit-map functions          */
#include <os2.h>

BOOL fSuccess;                  /* success indicator      */
HBITMAP hbm;                    /* bit-map handle         */
PSIZEL psizlBitmapDimension;    /* size of bit map       */
LONG lWidth;                    /* width of bit map       */

fSuccess = GpiQueryBitmapDimension(hbm, &psizlBitmapDimension);

/* if successful, assign value of bit-map width */
if (fSuccess == TRUE)
    lWidth = psizlBitmapDimension.cx;
```

GpiQueryBitmapInfoHeader – Query Bit-Map Info Header

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryBitmapInfoHeader (HBITMAP hbm, PBITMAPINFOHEADER2 pbmp2Data)

This function returns information about a bit map identified by the bit-map handle.

Parameters

hbm (HBITMAP) – input
Bit-map handle.

pbmp2Data (PBITMAPINFOHEADER2) – input/output
Bit-map information header.

This is a structure, that on return, is filled with data for the specified bit map.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HBITMAP

An invalid bit-map handle was specified.

PMERR_HBITMAP_BUSY

An internal bit map busy error was detected. The bit map was locked by one thread during an attempt to access it from another thread.

Remarks

The *cbFix* field of the BITMAPINFOHEADER2 structure must be set by the application before performing this function.

Note: This function should be used in preference to the GpiQueryBitmapParameters function.

GpiQueryBitmapInfoHeader — Query Bit-Map Info Header

Example Code

This example uses GpiQueryBitmapInfoHeader to return information about a bit map identified by the bit-map handle; if successful, it uses this information to create a new bit map via GpiCreateBitmap.

```
#define INCL_GPIBITMAPS          /* Bit-map functions          */
#include <os2.h>

HPS      hps;          /* presentation-space handle      */
BOOL     fSuccess;     /* success indicator              */
HBITMAP  hbm;          /* bit-map handle                 */
HBITMAP  hbmNew;       /* bit-map handle                 */
BITMAPINFOHEADER2 pbmp2Data; /* Bit-map information header    */
PBYTE pb;             /* address of bit-map image data in resource */

/* set size of info structure */
pbmp2Data.cbFix = 16L;

fSuccess = GpiQueryBitmapInfoHeader(hbm, &pbmp2Data);

/* use information to create bit map */
hbmNew = GpiCreateBitmap(hps,          /* presentation space      */
                        &pbmp2Data,    /* bit-map information header */
                        CBM_INIT,      /* initialize the bit map  */
                        pb,            /* bit-map data            */
                        (PBITMAPINFO2)&pbmp2Data);
                        /* bit-map information table */
```

GpiQueryBitmapHandle – Query Bit-Map Handle

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI or INCL_PM */
```

HBITMAP GpiQueryBitmapHandle (HPS hps, LONG lLcid)

This function returns the handle of the bit map currently tagged with the specified local identifier (lLcid).

Parameters

hps (HPS) – input
Presentation-space handle.

lLcid (LONG) – input
Local identifier.

Returns

Bit-map handle:

≠0 Bit-map handle

GPI_ERROR Error.

Possible returns from GetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_SETID	An invalid setid parameter was specified.
PMERR_ID_HAS_NO_BITMAP	No bit map was tagged with the setid specified on a GpiQueryBitmapHandle call.

Remarks

An error is raised if a bit map is not currently tagged with the specified lLcid.

Related Functions

- GpiSetBitmapId

Example Code

This example uses GpiQueryBitmapHandle to return the handle of the bit map currently tagged with the specified local identifier (lLcid) set by GpiSetBitmapId.

```
#define INCL_GPIBITMAPS          /* Bit-map functions          */
#include <os2.h>

HBITMAP hbm;                    /* bit-map handle          */
HPS hps;                        /* presentation-space handle */
LONG lLcid;                     /* local identifier         */

hbm = GpiQueryBitmapHandle(hps, lLcid);
```

GpiQueryBitmapParameters –

Query Bit-Map Parameters

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryBitmapParameters (HBITMAP hbm, PBITMAPINFOHEADER pbmpData)

This function returns information about a bit map identified by the bit-map handle.

Parameters

hbm (HBITMAP) – input
Bit-map handle.

pbmpData (PBITMAPINFOHEADER) – input/output
Bit-map information header.

This is a structure, that on return, is filled with data for the specified bit map. The structure includes the elements (width, height, planes, bitcount) of a bit-map information table.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from GetLastError

PMERR_INV_HBITMAP

An invalid bit-map handle was specified.

PMERR_HBITMAP_BUSY

An internal bit map busy error was detected. The bit map was locked by one thread during an attempt to access it from another thread.

Remarks

The *cbFix* field of the BITMAPINFOHEADER structure must be set by the application before performing this function.

Related Functions

- GpiCreateBitmap

GpiQueryBitmapParameters – Query Bit-Map Parameters

Example Code

This example uses GpiQueryBitmapParameters to return information about a bit map identified by the bit-map handle; if successful, it assigns the width field to a variable.

```
#define INCL_GPIBITMAPS          /* Bit-map functions          */
#include <os2.h>

BOOL    fSuccess;              /* success indicator          */
HBITMAP hbm;                   /* bit-map handle             */
BITMAPINFOHEADER pbmpData; /* bit-map information header */
USHORT usWidth;                /* width of bit map           */

/* set size of info structure */
pbmpData.cbFix = sizeof(BITMAPINFOHEADER);

fSuccess = GpiQueryBitmapParameters(hbm, &pbmpData);

/* if successful, assign value of bit-map width */
if (fSuccess == TRUE)
    usWidth = pbmpData.cx;
```

GpiQueryBoundaryData – Query Boundary Data

#define INCL_GPICORRELATION /* Or use INCL_GPI or INCL_PM */

BOOL GpiQueryBoundaryData (HPS hps, PRECTL prclBoundary)

This function returns the boundary data.

Parameters

- hps** (HPS) – input
Presentation-space handle.
- prclBoundary** (PRECTL) – output
Boundary data.

A rectangle structure in which the boundary data is returned, containing the following fields:

- xmin** Lowest x value found
ymin Lowest y value found
xmax Highest x value found
ymax Highest y value found.

Returns

- Success indicator:
- TRUE** Successful completion
FALSE Error occurred.

Possible returns from WinGetLastError

- | | |
|----------------------------------|---|
| PMERR_INV_HPS | An invalid presentation-space handle was specified. |
| PMERR_PS_BUSY | An attempt was made to access the presentation space from more than one thread simultaneously. |
| PMERR_COORDINATE_OVERFLOW | An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large. |
| PMERR_INV_DC_TYPE | An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context. |

Remarks

This function returns the boundary data set upon completion of the last boundary calculation. Boundary data is returned as the coordinates in model space.

Boundary data is inclusive. A null boundary is indicated if xmin is greater than xmax, or if ymin is greater than ymax. After GpiResetBoundaryData, xmin and ymin are the maximum positive numbers, and xmax and ymax are the maximum negative numbers.

GpiQueryBoundaryData – Query Boundary Data

Related Functions

- GpiResetBoundaryData
- GpiSetDrawControl

Example Code

This example uses the GpiQueryBoundaryData function to retrieve the rectangle enclosing the output. The boundary data is then used to draw a border around the output.

```
#define INCL_GPICORRELATION
#define INCL_GPIPRIMITIVES      /* GPI primitive functions */
#define INCL_GPICONTROL         /* GPI control Functions */
#include <os2.h>

HPS hps;                        /* presentation space handle */
POINTL ptlStart = { 0, 0 };    /* first vertex */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices */
RECTL rcl;                      /* rectangle */

GpiSetDrawControl(hps,
    DCTL_BOUNDARY, DCTL_ON);    /* accumulate boundary data */

GpiMove(hps, &ptlStart);        /* produce output */
GpiPolyLine(hps, 3L, ptlTriangle);

GpiQueryBoundaryData(hps, &rcl); /* copy boundary data to rcl */
if (rcl.xLeft < rcl.xRight) {   /* verify output exists*/
    ptlStart.x = rcl.xLeft; ptlStart.y = rcl.yBottom;
    GpiMove(hps, &ptlStart);    /* move to lower-right corner */
    ptlStart.x = rcl.xRight; ptlStart.y = rcl.yTop;
    GpiBox(hps, DRO_OUTLINE, &ptlStart, 0L, 0L); /* draw border */
}
```


GpiQueryCharAngle – Query Character Angle

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryCharAngle (HPS hps, PGRADIENTL pgradlAngle)

This function returns the current value of the character baseline angle.

Parameters

hps (HPS) – input

Presentation-space handle.

pgradlAngle (PGRADIENTL) – output

Baseline angle.

A point, relative to (0,0), that defines the character baseline angle vector.

If the character angle is currently set to the default value, (0,0) is returned.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE

An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE

An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Related Functions

- GpiQueryAttrs
- GpiSetCharAngle

GpiQueryCharAngle – Query Character Angle

Example Code

This example uses GpiQueryCharAngle to return the current value of the character baseline angle; if successful, it places the x component in a variable.

```
#define INCL_GPIPRIMITIVES    /* Primitive functions    */
#include <os2.h>

BOOL fSuccess;               /* success indicator    */
HPS  hps;                    /* Presentation-space handle */
LONG  lxComponent;           /* x component of baseline angle */
GRADIENTL pgradlAngle; /* Baseline angle */

fSuccess = GpiQueryCharAngle(hps, &pgradlAngle);

if (fSuccess == TRUE)
    lxComponent = pgradlAngle.x;
```

GpiQueryCharBox — Query Character Box

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryCharBox (HPS hps, PSIZEF pszfxSize)

This function returns the current value of the character box attribute, as set by the GpiSetCharBox function.

Parameters

hps (HPS) — input
Presentation-space handle.

pszfxSize (PSIZEF) — output
Character-box size.

If the character box is currently set to the default, the default size is returned. This is the size returned by DevQueryCaps (CAPS_GRAPHICS_CHAR_WIDTH and CAPS_GRAPHICS_CHAR_HEIGHT), converted to presentation page space.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from GetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

In general, this function does not return the same box as GpiQueryTextBox for an average-sized character. For outline fonts the character-box attribute is mapped to a particular font dimension related to the point size, for raster fonts it does not correspond to any font metric. (See GpiSetCharMode).

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Related Functions

- DevQueryCaps
- GpiQueryAttrs
- GpiSetCharBox

GpiQueryCharBox – Query Character Box

Example Code

This example uses GpiQueryCharBox to return the current value of the character box attribute, as set by the GpiSetCharBox call; if successful, places the box width in a variable.

```
#define INCL_GPIPRIMITIVES    /* Primitive functions    */
#include <os2.h>

BOOL fSuccess;                /* success indicator    */
HPS  hps;                     /* Presentation-space handle */
SIZEF psizfxSize;             /* Character-box size    */
FIXED lWidth;                 /* character box width    */

fSuccess = GpiQueryCharBox(hps, &psizfxSize);

if (fSuccess == TRUE)
    lWidth = psizfxSize.cx;
```

GpiQueryCharBreakExtra — Query Character Break Extra

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryCharBreakExtra (HPS hps, PFIXED pfxBreakExtra)

This function returns the current value of the character-break-extra attribute, as set by the GpiSetCharBreakExtra function.

Parameters

hps (HPS) — input

Presentation-space handle.

pfxBreakExtra (PFIXED) — output

Character-break-extra attribute value.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from GetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE

An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Example Code

This example uses GpiQueryCharBreakExtra to return the current value of the character-break-extra attribute, as set by the GpiSetCharBreakExtra call.

```
#define INCL_GPIPRIMITIVES /* Primitive functions */
#include <os2.h>

BOOL fSuccess; /* success indicator */
HPS hps; /* Presentation-space handle */
FIXED pfxBreakExtra; /* Character-break-extra attribute value*/

fSuccess = GpiQueryCharBreakExtra(hps, &pfxBreakExtra);
```

GpiQueryCharDirection – Query Character Direction

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryCharDirection (HPS hps)

This call returns the current value of the character direction attribute, as set by the GpiSetCharDirection function.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Character direction:

CHDIRN_DEFAULT	Default
>0	Character direction
CHDIRN_ERROR	Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This call is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Related Functions

- GpiQueryAttrs
- GpiSetCharDirection

Example Code

This example uses GpiQueryCharDirection to return the current value of the character direction attribute, as set by the GpiSetCharDirection call.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions      */
#include <os2.h>

LONG lDirection;               /* character direction (or error) */
HPS hps;                       /* Presentation-space handle      */

lDirection = GpiQueryCharDirection(hps);
```

GpiQueryCharExtra — Query Character Extra

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryCharExtra (HPS hps, PFIXED pfxExtra)

This function returns the current value of the character-extra attribute, as set by the GpiSetCharExtra function.

Parameters

hps (HPS) — input
Presentation-space handle.

pfxExtra (PFIXED) — output
Character-extra attribute value.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Example Code

This example uses GpiQueryCharExtra to return the current value of the character-extra attribute, as set by the GpiSetCharExtra call.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions      */
#include <os2.h>

BOOL fSuccess;                 /* success indicator      */
HPS  hps;                     /* Presentation-space handle */
FIXED pfxExtra;               /* Character-extra attribute value. */

fSuccess = GpiQueryCharExtra(hps, &pfxExtra);
```

GpiQueryCharMode – Query Character Mode

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryCharMode (HPS hps)

This function returns the current value of the character-mode attribute, as set by the GpiSetCharMode function.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Character mode:

CM_DEFAULT Default

>0 Character mode

CM_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE

An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE

An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Related Functions

- GpiQueryAttrs
- GpiSetCharMode

Example Code

This example uses GpiQueryCharMode to return the current value of the character mode attribute, as set by the GpiSetCharMode call.

```
#define INCL_GPIPRIMITIVES /* Primitive functions */
#include <os2.h>

LONG lMode; /* character mode attribute */
HPS hps; /* Presentation-space handle */

lMode = GpiQueryCharMode(hps);
```


GpiQueryCharSet — Query Character Set

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

```
LONG GpiQueryCharSet (HPS hps)
```

This function returns the character-set local identifier (lcid), as set by the GpiSetCharSet function.

Parameters

hps (HPS) — input
Presentation-space handle.

Returns

Character-set local identifier:

LCID_DEFAULT Default

>0 Local identifier

LCID_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Related Functions

- GpiQueryAttrs
- GpiSetCharSet

Example Code

This example uses GpiQueryCharSet to return the character-set local identifier (lcid), as set by the GpiSetCharSet call.

```
#define INCL_GPIPRIMITIVES /* Primitive functions */
#include <os2.h>

LONG lLcid; /* character set lcid (or error) */
HPS hps; /* Presentation-space handle */

lLcid = GpiQueryCharSet(hps);
```

GpiQueryCharShear – Query Character Shear

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryCharShear (HPS hps, PPOINTL pptlShear)

This function returns the value of the current character-shear angle, as set by the GpiSetCharShear function.

Parameters

hps (HPS) – input
Presentation-space handle.

pptlShear (PPOINTL) – output
Character shear.

A point, relative to (0,0), that defines the character shear vector.

If the character shear is currently set to the default, (0,1) is returned.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE

An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE

An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Related Functions

- GpiQueryAttrs
- GpiSetCharShear

GpiQueryCharShear – Query Character Shear

Example Code

This example uses GpiQueryCharShear to return the value of the current character-shear angle, as set by the GpiSetCharShear call; if successful, it assigns the x coordinate of the returned vector to a variable.

```
#define INCL_GPIPRIMITIVES    /* Primitive functions      */
#include <os2.h>

BOOL fSuccess;               /* success indicator      */
HPS hps;                     /* Presentation-space handle */
POINTL pptlShear;            /* character shear         */
LONG lxCoord;                /* shear angle vector x coordinate */

fSuccess = GpiQueryCharShear(hps, &pptlShear);

if (fSuccess == TRUE)
    lxCoord = pptlShear.x;
```

GpiQueryCharStringPos – Query Character String Positions

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryCharStringPos (HPS hps, ULONG flOptions, LONG lCount, PCH pchString, PLONG alXIncrements, PPOINTL aptlPositions)

This function processes a string as if it is being drawn under the current character attributes using GpiCharStringPos, and returns the positions in the string at which each character would be drawn.

Parameters

hps (HPS) – input
Presentation-space handle.

flOptions (ULONG) – input
Option flag:

CHS_VECTOR Increments vector supplied (*alXIncrements*). If 0, *alXIncrements* is ignored.

lCount (LONG) – input
Length of the string.

pchString (PCH) – input
Character string to be examined.

alXIncrements (PLONG) – input
Vector of x increment values.

These are signed values in world coordinates. Any negative values are treated as if they were 0. This parameter is ignored if CHS_VECTOR is not set.

aptlPositions (PPOINTL) – output
Array of points.

The positions of each character in world coordinates. The first point returned is the initial current position, and the last point is the new current position if the string has been drawn.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE

An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_CHAR_POS_OPTIONS

An invalid options parameter was specified with GpiCharStringPos or GpiCharStringPosAt.

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

PMERR_INV_COORDINATE

An invalid coordinate value was specified.

GpiQueryCharStringPos — Query Character String Positions

PMERR_COORDINATE_OVERFLOW

An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

PMERR_INV_DC_TYPE

An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

A vector of increments can be specified, allowing control over the positioning of each character after the first. These are distances measured in world coordinates (along the baseline for left-to-right and right-to-left character directions, and along the shearline for top-to-bottom and bottom-to-top). The *i*'th increment is the distance of the reference point of the (*i* + 1)'th character from the reference point of the *i*'th. The last increment may be needed to update current position.

These increments, if specified, set the widths of each character.

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Related Functions

- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiQueryCharStringPosAt
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear

Example Code

This example calls the GpiQueryCharStringPos function to determine the location of each character in the string. Vector increments are not used.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions */
#include <os2.h>

HPS hps;                        /* presentation space handle */
CHAR szString[] = "Sample string";
POINTL aptl[sizeof(szString) + 1];

GpiQueryCharStringPos(hps,      /* presentation-space handle */
    0L,                        /* does not use vector increments */
    sizeof(szString),          /* number of characters in string */
    szString,                  /* character string */
    NULL,                      /* no vector increments */
    aptl);                     /* array of structures for points */
```

GpiQueryCharStringPosAt – Query Character String Positions At

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

```
BOOL GpiQueryCharStringPosAt (HPS hps, PPOINTL pptlStart, ULONG flOptions,  
                             LONG lCount, PCH pchString, PLONG alXIncrements,  
                             PPOINTL aptlPositions)
```

This function processes a string as if it is being drawn under the current character attributes using GpiCharStringPosAt, and returns the positions in the string at which each character would be drawn.

Parameters

hps (HPS) – input
Presentation-space handle.

pptlStart (PPOINTL) – input
Starting position.

flOptions (ULONG) – input
Option flags:

CHS_VECTOR Increments vector supplied (*alXincrements*). If 0, *alXincrements* is ignored.

lCount (LONG) – input
Length of the string.

pchString (PCH) – input
Character string to be examined.

alXIncrements (PLONG) – input
Vector of x increment values.

These are signed values in world coordinates. Any negative values are treated as if they were 0. This parameter is ignored if **CHS_VECTOR** is not set.

aptlPositions (PPOINTL) – output
Array of points, in which the positions of each character in world coordinates are returned.

The first point returned is the initial current position, and the last point is the new current position if the string has been drawn.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE

An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_CHAR_POS_OPTIONS

An invalid options parameter was specified with GpiCharStringPos or GpiCharStringPosAt.

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

GpiQueryCharStringPosAt — Query Character String Positions At

PMERR_INV_COORDINATE	An invalid coordinate value was specified.
PMERR_COORDINATE_OVERFLOW	An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

A vector of increments can be specified, allowing control over the positioning of each character after the first. These are distances measured in world coordinates (along the baseline for left-to-right and right-to-left character directions, and along the shearline for top-to-bottom and bottom-to-top). The *i*'th increment is the distance of the reference point of the (*i* + 1)'th character from the reference point of the *i*'th. The last increment may be needed to update current position.

These increments, if specified, set the widths of each character.

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Related Functions

- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiQueryCharStringPos
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear

Example Code

This example uses the GpiQueryCharStringPosAt function to determine the location of each character in the string. Vector increments are not used.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions */
#include <os2.h>

HPS hps;                        /* presentation space handle */
POINTL ptlStart = { 100, 100 };
POINTL aptl[12];

GpiQueryCharStringPosAt(hps,    /* presentation-space handle */
    &ptlStart,                  /* starting point for string */
    0L,                        /* do not use vector increments */
    11L,                       /* 11 characters in string */
    "This string",             /* character string */
    NULL,                      /* no vector increments */
    aptl);                     /* array of structures for points */
```

```
#define INCL_GPIREGIONS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryClipBox (HPS hps, PRECTL prclBound)

This function returns the dimensions of the tightest rectangle which completely encloses the intersection of *all* the clipping definitions.

Parameters

hps (HPS) – input

Presentation-space handle.

prclBound (PRECTL) – output

Bounding rectangle.

The coordinates of the bounding rectangle, in world coordinates.

Returns

Complexity and error indicators:

RGN_NULL	Null region
RGN_RECT	Rectangular region
RGN_COMPLEX	Complex region
RGN_ERROR	Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_COORDINATE_OVERFLOW

An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

Remarks

The clipping definitions include the combined effects of:

- Clip path
- Viewing limits
- Graphics field
- Clip region
- Visible region (windowing considerations).

Points on the borders of the rectangle returned are considered to be included within the rectangle. If the intersection is null, the rectangle returned has the right boundary less than the left, and the top boundary less than the bottom.

GpiQueryClipBox —

Query Clip Box

Example Code

This example uses GpiQueryClipBox to return the dimensions of the tightest rectangle which completely encloses the intersection of all the clipping definitions. The example queries the clip box and, if a rectangular region is returned, assigns the x coordinate of the lower left hand corner of the clip box region to a variable.

```
#define INCL_GPIREGIONS      /* Region functions      */
#include <os2.h>

LONG lComplexity;           /* complexity/error indicator */
HPS  hps;                  /* Presentation-space handle  */
RECTL prclBound;           /* bounding rectangle          */
LONG  lLwrLftxCoord;       /* lower left x coordinate of clip box */

lComplexity = GpiQueryClipBox(hps, &prclBound);

/* if returned region is a rectangle, assign lower left x coordinate */
if (lComplexity == RGN_RECT)
    lLwrLftxCoord = prclBound.xLeft;
```

GpiQueryClipRegion – Query Clip Region

```
#define INCL_GPIREGIONS /* Or use INCL_GPI or INCL_PM */
```

HRGN GpiQueryClipRegion (HPS hps)

This function returns the handle of the currently selected clip region.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Clip-region handle (if any):

NULLHANDLE Null handle (no region is selected)

HRGN_ERROR Error

Otherwise Clip region handle.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

Remarks

If there is no currently selected clip region, a null handle is returned.

Example Code

This example uses GpiQueryClipRegion to return the handle of the currently selected clip region.

```
#define INCL_GPIREGIONS      /* Region functions          */
#include <os2.h>

HPS    hps;                /* Presentation-space handle */
HRGN   hrgn;               /* clip region handle        */

hrgn = GpiQueryClipRegion(hps);
```

GpiQueryColor – Query Color

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

LONG GpiQueryColor (HPS hps)

This function returns the current value of the (character) color attribute, as set by the GpiSetColor call.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Color attribute:

CLR_ERROR Error

CLR_DEFAULT Default

Otherwise Color index.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryColor – Query Color

Example Code

This example uses GpiQueryColor to return the current value of the (character) color attribute, then sets the color to red by calling GpiSetColor. When finished with red, the color is set back to its original value.

```
#define INCL_GPIPRIMITIVES    /* Primitive functions    */
#include <os2.h>

LONG    lColor;              /* current character color (or error) */
HPS     hps;                /* Presentation-space handle          */
HPS     GEhps;

/* query current color */
lColor = GpiQueryColor(hps);

/* set color to red */
GpiSetColor(GEhps, CLR_RED);
.
.
/* restore to original color */
GpiSetColor(GEhps, lColor);
```

GpiQueryColorData —

Query Color Data

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryColorData (HPS hps, LONG ICount, PLONG alArray)

Information about the current logical color table or the selected palette is returned by this function.

Parameters

hps (HPS) — input

Presentation-space handle.

ICount (LONG) — input

Number of elements.

Number of elements supplied in *alArray*.

alArray (PLONG) — output

Array.

On return this array contains:

Array[QCD_LCT_FORMAT] Format of loaded color table if any. One of the following values is returned:

LCOLF_DEFAULT Default color table is in force.

LCOLF_INDRGB Color table loaded which provides translation from index to RGB.

LCOLF_RGB Color index = RGB.

LCOLF_PALETTE Palette is selected.

Array[QCD_LCT_LOINDEX] Smallest color index in the color table or palette ; always zero for color tables.

Array[QCD_LCT_HIINDEX] Largest color index in the color table or palette ; never less than 15 for color tables.

Array[QCD_LCT_OPTIONS] Color table or palette option. Zero or more of the following are returned:

LCOL_PURECOLOR No color dithering (color table or selected palette).

LCOL_OVERRIDE_DEFAULT_COLORS Override for applications that need the full hardware palette (selected palette only)

The array elements are numbered consecutively, starting with **Array[QCD_LCT_FORMAT]**. The element number constants start with 0. (See the appropriate Bindings Reference.)

Information is returned only for the number of elements supplied. Any extra elements supplied, beyond those described above, are set to zero by the system.

GpiQueryColorData — Query Color Data

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

Example Code

This example uses the GpiQueryColorData function to retrieve the smallest color-table index. The GpiQueryLogColorTable function is then used to retrieve the RGB color value for this index.

```
#define INCL_GPILOGCOLORTABLE    /* Color Table functions        */
#include <os2.h>

HPS hps;                        /* presentation space handle        */
LONG a1Data[3];                /* information array                */
LONG a1Color[1];               /* information array                */

GpiQueryColorData(hps, 3L, a1Data);
GpiQueryLogColorTable(hps, 0L, a1Data[QCD_LCT_LOINDEX],
                                 1L, a1Color);
```

GpiQueryColorIndex —

Query Color Index

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryColorIndex (HPS hps, ULONG ulOptions, LONG IRgbColor)

This function returns the color index of the device color that is closest to the specified RGB color representation for the device connected to the specified presentation space.

Parameters

hps (HPS) — input
Presentation-space handle.

ulOptions (ULONG) — input
Options:
Reserved, and must be zero.

IRgbColor (LONG) — input
Specifies a color in RGB terms.

Returns

Color index providing closest match to the specified color:

≥0 Color index

GPI_ALTEERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_COLOR_OPTIONS	An invalid options parameter was specified with a logical color table or color query function.
PMERR_INV_RGBCOLOR	An invalid rgb color parameter was specified with GpiQueryNearestColor or GpiQueryColor

Remarks

If an RGB logical color table has been loaded, this call returns the same RGB color that is passed to it.

GpiQueryColorIndex – Query Color Index

Example Code

This example uses GpiQueryColorIndex to return the color index of the device color that is closest to the specified RGB color representation for the device connected to the specified presentation space.

```
#define INCL_GPILOGCOLORTABLE  /* Color Table functions      */
#include <os2.h>

LONG  lIndex;          /* closest match color index      */
HPS   hps;             /* Presentation-space handle      */
ULONG ulOptions;       /* options                        */
LONG  lRgbColor;       /* color in RGB terms            */

/* reserved; set to 0 */
ulOptions = 0L;

/* color to find index for */
lRgbColor = (PC_RESERVED*16777216) + (0*65536) + (0*256) + 1;

lIndex = GpiQueryColorIndex(hps, ulOptions, lRgbColor);
```


GpiQueryCp – Query Code Page

```
#define INCL_GPILCIDS /* Or use INCL_GPI or INCL_PM */
```

ULONG GpiQueryCp (HPS hps)

This function returns the currently selected graphics code page.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Code page:

GPI_ERROR Error

Otherwise Code page.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space
from more than one thread simultaneously.

Remarks

The code page identity returned is the one that is set by GpiSetCp (or defaulted when the presentation space is first created). This is the code page of the default font, not the currently-selected font, found from GpiQueryFontMetrics.

Example Code

This example uses GpiQueryCp to return the currently selected graphics code page.

```
#define INCL_GPILCIDS                      /* Font functions                      */
#include <os2.h>

ULONG    ulCodePage;                      /* code page (or error)                      */
HPS    hps;                               /* Presentation-space handle                      */

ulCodePage = GpiQueryCp(hps);
```

GpiQueryCurrentPosition — Query Current Position

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryCurrentPosition (HPS hps, PPOINTL pptlPoint)

This function returns the value of current position.

Parameters

hps (HPS) — input
Presentation-space handle.

pptlPoint (PPOINTL) — output
Current position.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Example Code

This example uses GpiQueryCurrentPosition to return the value of the current position and assigns the x coordinate to a variable.

```
#define INCL_GPIPRIMITIVES /* Primitive functions */
#include <os2.h>

BOOL fSuccess; /* success indicator */
HPS hps; /* Presentation-space handle */
POINTL pptlPoint; /* current position */
LONG lxCoord; /* current position x coordinate */

fSuccess = GpiQueryCurrentPosition(hps, &pptlPoint);

if (fSuccess == TRUE)
    lxCoord = pptlPoint.x;
```

GpiQueryDefArcParams – Query Default Arc Parameters

```
#define INCL_GPIDEFAULTS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryDefArcParams (HPS hps, PARCPARAMS parcpArcParams)

This function returns the default values of the arc parameters, as set by the GpiSetDefArcParams function.

Parameters

hps (HPS) – input
Presentation-space handle.

parcpArcParams (PARCPARAMS) – output
Default arc parameters.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

Example Code

This example uses GpiQueryDefArcParams to return the default values of the arc parameters, as set by the GpiSetDefArcParams call, and assign a variable to the P coefficient if the query succeeds.

```
#define INCL_GPIDEFAULTS /* Default functions */
#include <os2.h>

BOOL fSuccess; /* success indicator */
HPS hps; /* Presentation-space handle */
PARCPARAMS parcpArcParams; /* Arc parameters */
LONG lPcoefficient; /* p coefficient of arc definition */

fSuccess = GpiQueryDefArcParams(hps, &parcpArcParams);

/* if successful, assign value of P coefficient */
if (fSuccess == TRUE)
    lPcoefficient = parcpArcParams.lP;
```

GpiQueryDefAttrs – Query Default Attributes

```
#define INCL_GPIDEFAULTS /* Or use INCL_GPI or INCL_PM */
```

**BOOL GpiQueryDefAttrs (HPS hps, LONG IPrimType, ULONG flAttrMask,
PBUNDLE ppbunAttrs)**

This function returns default attribute values for the specified primitive type.

Parameters

hps (HPS) – input
Presentation-space handle.

IPrimType (LONG) – input
Primitive type.

This is the type of primitive for which default attribute values are to be queried, as follows:

PRIM_LINE	Line and arc primitives
PRIM_CHAR	Character primitives
PRIM_MARKER	Marker primitives
PRIM_AREA	Area primitives
PRIM_IMAGE	Image primitives.

flAttrMask (ULONG) – input
Attributes mask.

Each flag that is set indicates that the default value of the corresponding attribute is to be returned in the *ppbunAttrs* buffer.

If all flags in *flAttrMask* are 0, the *ppbunAttrs* buffer address is not used.

ppbunAttrs (PBUNDLE) – output
Attributes.

ppbunAttrs is a buffer in which is returned the default value of each attribute for which the *flAttrMask* flag is set, in the order specified in *GpiSetAttrs* for the particular primitive type.

Only data for attributes for which the appropriate flag in *flAttrMask* is set is updated, so *ppbunAttrs* need only be large enough for the highest offset attribute to be returned (see *GpiSetAttrs*).

Returns

Success indicator:

TRUE	Successful completion
FALSE	Error occurred.

Possible returns from *WinGetLastError*

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_PRIMITIVE_TYPE	An invalid primitive type parameter was specified with <i>GpiSetAttrs</i> or <i>GpiQueryAttrs</i> .
PMERR_UNSUPPORTED_ATTR	An unsupported attribute was specified in the attrmask with <i>GpiSetAttrs</i> or <i>GpiQueryAttrs</i> .

GpiQueryDefAttrs — Query Default Attributes

Remarks

The parameters returned by this function may be used to reinstate exactly the same default attribute values as are queried, using GpiSetDefAttrs.

Example Code

This example uses GpiQueryDefAttrs to return the default color and mix attribute values for the primitive line and arc types and, if successful, uses the values to reinstate the default attributes via the DosSetDefAttrs API.

```
#define INCL_GPIDEFAULTS      /* Default functions      */
#include <os2.h>

BOOL fSuccess;               /* success indicator      */
HPS hps;                     /* Presentation-space handle */
LONG lPrimType;              /* primitive type          */
ULONG flAttrMask;            /* attributes mask          */
LINEBUNDLE ppbunAttrs; /* Attributes              */

/* request line/arc primitive values */
lPrimType = PRIM_LINE;

/* request values for color, mix attributes */
flAttrMask = LBB_COLOR | LBB_MIX_MODE;

fSuccess = GpiQueryDefAttrs(hps, lPrimType, flAttrMask,
                           &ppbunAttrs);

/* if successful, reinstate default color and mix attributes */
if (fSuccess == TRUE)
    fSuccess = GpiSetDefAttrs(hps, lPrimType, flAttrMask,
                             &ppbunAttrs);
```

GpiQueryDefaultViewMatrix – Query Default View Matrix

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryDefaultViewMatrix (HPS hps, LONG ICount, PMATRIXLF pmatlfArray)

This function returns the current default viewing transform; see GpiSetDefaultViewMatrix.

Parameters

hps (HPS) – input
Presentation-space handle.

ICount (LONG) – input
Number of elements.

The number of elements to be returned in *pmatlfArray*; must be in the range 0 through 9. If 0 is specified, no matrix elements are returned.

pmatlfArray (PMATRIXLF) – output
Transform matrix.

An array into which the elements of the default viewing transform matrix are returned.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT An invalid length or count parameter was specified.

GpiQueryDefaultViewMatrix — Query Default View Matrix

Example Code

This example uses GpiQueryDefaultViewMatrix to return the default viewing transform and, if successful, defines - via DosSetDefaultViewMatrix - the returned value as the new default transform.

```
#define INCL_GPITRANSFORMS    /* Transform functions          */
#include <os2.h>

BOOL fSuccess;               /* success indicator          */
HPS  hps;                    /* Presentation-space handle  */
LONG lCount;                 /* number of elements         */
MATRIXLF pmatlfArray;        /* transform matrix           */
LONG  lOptions;              /* set default options        */

lCount = 1; /* examine only first element of transform matrix */

fSuccess = GpiQueryDefaultViewMatrix(hps, lCount, &pmatlfArray);

/* set default to returned transform */
if (fSuccess == TRUE)
{
    lOptions = TRANSFORM_REPLACE;
    fSuccess = GpiSetDefaultViewMatrix(hps, lCount, &pmatlfArray,
                                       lOptions);
}
```

GpiQueryDefCharBox – Query Default Graphics Character Box

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

```
BOOL GpiQueryDefCharBox (HPS hps, PSIZEL pszlSize)
```

This function returns the size of the default graphics character box in world coordinates.

Parameters

hps (**HPS**) – input
Presentation-space handle.

pszlSize (**PSIZEL**) – output
Default character-box size.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .

Remarks

The values returned are the same as the initial default value of the character-box attribute. See GpiSetCharBox for further information.

GpiQueryDefCharBox — Query Default Graphics Character Box

Example Code

This example uses GpiQueryDefCharBox to query the initial size of the default graphics character box in world coordinates and, if the query succeeds, resets the current size back to this initial default value via GpiSetCharBox (note the required transformation from LONG to FIXED using the MAKEFIXED macro).

```
#define INCL_GPIPRIMITIVES    /* Primitive functions      */
#include <os2.h>

BOOL fSuccess;               /* success indicator      */
HPS hps;                     /* Presentation-space handle */
SIZEL psizlSize;             /* default character-box size */
SIZEF psizfxSize;           /* new character-box size   */

fSuccess = GpiQueryDefCharBox(hps, &psizlSize);

/* if successful, set current box size to initial default value */
if (fSuccess == TRUE)
{
    psizfxSize.cx = MAKEFIXED(psizlSize.cx, 0x0000);
    psizfxSize.cy = MAKEFIXED(psizlSize.cy, 0x0000);
    GpiSetCharBox(hps, &psizfxSize);
}
```

GpiQueryDefTag – Query Default Tag

```
#define INCL_GPIDEFAULTS /* Or use INCL_GPI or INCL_PM */
```

```
BOOL GpiQueryDefTag (HPS hps, PLONG plTag)
```

This function returns the default value of the tag identifier, as set by the GpiSetDefTag function.

Parameters

hps (HPS) – input
Presentation-space handle.

plTag (PLONG) – output
Default tag identifier.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION An attempt was made to issue a function that is invalid in a micro presentation space.

Example Code

This example uses GpiQueryDefTag to return the default value of the tag identifier, as set by the GpiSetDefTag call.

```
#define INCL_GPIDEFAULTS      /* Default functions      */
#include <os2.h>

BOOL fSuccess;      /* success indicator      */
HPS hps;            /* Presentation-space handle      */
LONG plTag;         /* default tag identifier      */

fSuccess = GpiQueryDefTag(hps, &plTag);
```

GpiQueryDefViewingLimits – Query Default Viewing Limits

```
#define INCL_GPIDEFAULTS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryDefViewingLimits (HPS hps, PRECTL prclLimits)

This function returns the default value of the viewing limits, as set by the GpiSetDefViewingLimits function.

Parameters

hps (HPS) – input
Presentation-space handle.

prclLimits (PRECTL) – output
Default viewing limits.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.

Example Code

This example uses GpiQueryDefViewingLimits to return the default value of the viewing limits, as set by the GpiSetDefViewingLimits and, if the query succeeds, assigns a variable to the x coordinate of the lower left hand corner of the viewing limits rectangle.

```
#define INCL_GPIDEFAULTS      /* Default functions      */
#include <os2.h>

BOOL fSuccess;               /* success indicator      */
HPS  hps;                    /* Presentation-space handle */
RECTL prclLimits;            /* default viewing limits  */
LONG  lLwrLftxCoord;         /* lower left x coordinate of limit */

fSuccess = GpiQueryDefViewingLimits(hps, &prclLimits);

/* if successful, assign lower left x coordinate of viewing limit */
if (fSuccess == TRUE)
    lLwrLftxCoord = prclLimits.xLeft;
```

```
#define INCL_GPICONTROL /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

HDC GpiQueryDevice (HPS hps)

This function returns the handle of the currently associated device context.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Device-context handle:

HDC_ERROR Error

NULLHANDLE No device context is currently associated

Otherwise Device context handle.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_INV_HDC An invalid device-context handle or (micro presentation space) presentation-space handle was specified.

Example Code

This example uses the GpiQueryDevice function to retrieve a device-context handle for the presentation space of the desktop window. The handle is used in the DevQueryCaps function to determine the width and height of the Presentation Manager screen.

```
#define INCL_GPICONTROL      /* GPI control Functions      */
#define INCL_WINWINDOWMGR   /* Window Manager Functions */
#define INCL_DEV             /* Device Function definitions */
#include <os2.h>

HPS hps;                    /* presentation space handle */
HDC hdc;                    /* device context handle      */
LONG lWidth, lHeight;

hps = WinGetScreenPS(HWND_DESKTOP);
hdc = GpiQueryDevice(hps);
DevQueryCaps(hdc, CAPS_WIDTH, 1L, &lWidth);
DevQueryCaps(hdc, CAPS_HEIGHT, 1L, &lHeight);
```

GpiQueryDeviceBitmapFormats —

Query Device Bit-Map Formats

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryDeviceBitmapFormats (HPS hps, LONG ICount, PLONG alArray)

This function returns the formats of bit maps supported internally by the device driver.

Parameters

hps (HPS) — input

Presentation-space handle.

The associated device context defines the class of device for which formats are required. This must be either a memory device context or a device context for a device that supports raster operations.

ICount (LONG) — input

Number of elements.

Number of elements in *alArray* (must be an even number). For the complete set of formats returned, the value of this parameter must be at least double the number of device formats returned by DevQueryCaps.

alArray (PLONG) — output

Data array.

Array of elements that, on return, is set to pairs of (*cPlanes*, *cBitCount*) elements (see BITMAPINFOHEADER) for each supported format in turn. Any unused elements are set to 0.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from GetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

Remarks

An application can create, set, and query bit maps using any of the standard formats. Internally, however, these are converted by the device driver into one of the device internal formats if necessary. This is normally a smaller set than the standard set of bit-map formats.

The number of device bit-map formats can be found with DevQueryCaps (CAPS_BITMAP_FORMATS).

The first pair of (*cPlanes*, *cBitCount*) elements returned most closely matches the device.

This function must not be issued when there is no device context associated with the presentation space.

GpiQueryDeviceBitmapFormats – Query Device Bit-Map Formats

Example Code

This example uses the GpiQueryDeviceBitmapFormats function to retrieve bit-map formats for the screen and creates a screen-compatible bit map with GpiCreateBitmap.

```
#define INCL_GPIBITMAPS          /* GPI Bit-map functions */
#include <os2.h>

HPS      hps;          /* Target presentation-space handle */
LONG     lFormats[24]; /* Formats supported by the device */
HBITMAP  hbm;          /* Bit-map handle */
PBYTE    pb;           /* Bit-map image data */
BITMAPINFO2 pbmInfo;   /* Bit-map information table */

/* Get screen supportable formats */
GpiQueryDeviceBitmapFormats(hps, 24L, lFormats);

/*****
 * set bitmapinfo structure *
 *****/
pbmInfo.cbFix = 16L;
pbmInfo.cx = 100L;
pbmInfo.cy = 100L;
pbmInfo.cPlanes = (USHORT) lFormats[0] ;
pbmInfo.cBitCount = (USHORT) lFormats[1];

/* create bit map and return handle */
hbm = GpiCreateBitmap(hps,          /* presentation space */
                     (PBITMAPINFOHEADER2)&pbmInfo,
                     /* bit-map information header */
                     CBM_INIT,     /* initialize the bit map */
                     pb,           /* bit-map data */
                     &pbmInfo);   /* bit-map information table */
```

GpiQueryDrawControl – Query Draw Control

```
#define INCL_GPICONTROL /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryDrawControl (HPS hps, LONG IControl)

This function returns a drawing control as set by GpiSetDrawControl.

Parameters

hps (HPS) – input

Presentation-space handle.

IControl (LONG) – input

Control whose value is to be returned:

DCTL_ERASE Erase before draw

DCTL_DISPLAY Display

DCTL_BOUNDARY Accumulate boundary data

DCTL_DYNAMIC Draw dynamic segments

DCTL_CORRELATE Correlate.

Returns

Value of the control.

(See GpiSetDrawControl for details):

DCTL_OFF Off

DCTL_ON On

DCTL_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_DRAW_CONTROL

An invalid control parameter was specified with GpiSetDrawControl or GpiQueryDrawControl.

PMERR_INV_MICROPS_DRAW_CONTROL

A draw control parameter was specified with GpiSetDrawControl that is invalid in a micro presentation space.

GpiQueryDrawControl — Query Draw Control

Example Code

This example uses GpiQueryDrawControl to return the value for the Display drawing control as set by GpiSetDrawControl.

```
#define INCL_GPICONTROL /* Control functions */
#include <os2.h>

LONG lValue; /* value of the control */
HPS hps; /* Presentation-space handle */
LONG lControl; /* control value to be queried */

/* ask for Display control value */
lControl = DCTL_DISPLAY;

lValue = GpiQueryDrawControl(hps, lControl);
```


GpiQueryDrawingMode — Query Drawing Mode

```
#define INCL_GPICONTROL /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryDrawingMode (HPS hps)

This function returns the current drawing mode, as set by GpiSetDrawingMode.

Parameters

hps (HPS) — input
Presentation-space handle.

Returns

Drawing mode.

(See GpiSetDrawingMode for details):

>0 Drawing mode

DM_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION An attempt was made to issue a function that is invalid in a micro presentation space.

Example Code

This example uses GpiQueryDrawingMode to return the current drawing mode, as set by GpiSetDrawingMode.

```
#define INCL_GPICONTROL /* Control functions */
#include <os2.h>

LONG lMode; /* drawing mode */
HPS hps; /* Presentation-space handle */

lMode = GpiQueryDrawingMode(hps);
```

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryEditMode (HPS hps)

This function returns the current editing mode, as set by GpiSetEditMode.

Parameters

hps (HPS) — input
Presentation-space handle.

Returns

Current editing mode:

SEGEN_INSERT	Insert mode
SEGEN_REPLACE	Replace mode
SEGEN_ERROR	Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.

Remarks

This function can be issued in any drawing mode.

Example Code

This example uses GpiQueryEditMode to return the current editing mode, as set by GpiSetEditMode.

```
#define INCL_GPISEGEDITING /* Segment Editing functions */
#include <os2.h>

LONG lMode; /* editing mode */
HPS hps; /* Presentation-space handle */

lMode = GpiQueryEditMode(hps);
```

GpiQueryElement — Query Element

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryElement (HPS hps, LONG IOff, LONG IMaxLength, PBYTE pbData)

This function returns element content data.

Parameters

hps (HPS) — input

Presentation-space handle.

IOff (LONG) — input

Starting byte offset within the element.

IMaxLength (LONG) — input

Maximum length of data that can be returned.

pbData (PBYTE) — output

Element content data.

An area of *IMaxLength* bytes in which the element content data is to be returned.

Returns

Number of bytes returned:

≥ 0 Actual number of bytes returned

GPI_ALTERERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION

An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_NO_CURRENT_ELEMENT

An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open element.

PMERR_NOT_IN_RETAIN_MODE

An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to **retain**

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

PMERR_INV_IN_ELEMENT

An attempt was made to issue a function invalid inside an element bracket.

PMERR_INV_ELEMENT_OFFSET

An invalid off (offset) parameter was specified with GpiQueryElement.

PMERR_NO_CURRENT_SEG

An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open segment.

GpiQueryElement — Query Element

Remarks

Returns the element content (or part of the element content) for the element to which the element pointer currently points.

This function is only valid when the drawing mode (see GpiSetDrawingMode) is set to **retain** (not **draw-and-retain**), and a segment bracket is currently in progress.

This function is not valid within an element bracket.

Example Code

This example uses the GpiQueryElement function to retrieve the graphics-order data for an element.

```
#define INCL_GPISEGEDITING      /* GPI Segment Edit functions */
#include <os2.h>

HPS hps;           /* presentation space handle */
BYTE abElement[512]; /* element data buffer */

/* Move pointer to first element in segment. */

GpiSetElementPointer(hps, 1L);
GpiQueryElement(hps, /* presentation space */
                0L, /* start with first byte in element */
                512L, /* copy no more than 512 bytes */
                abElement); /* buffer to receive data */
```

GpiQueryElementPointer – Query Element Pointer

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryElementPointer (HPS hps)

This function returns the current element pointer.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Current element pointer:

≥0 Current element pointer

GPI_ALTError Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.
PMERR_NOT_IN_RETAIN_MODE	An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to retain
PMERR_NO_CURRENT_SEG	An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open segment.

Remarks

This function is only valid when the drawing mode (see GpiSetDrawingMode) is set to **retain** (not **draw-and-retain**), and a segment bracket is currently in progress.

GpiQueryElementPointer — Query Element Pointer

Example Code

This example uses GpiQueryElementPointer to return the current element pointer after setting the Draw mode to retain and beginning a graphics segment named 1.

```
#define INCL_GPISEGEDITING    /* Segment Editing functions */
#define INCL_GPICONTROL      /* Control functions */
#define INCL_GPISEGMENTS     /* Segment functions */
#include <os2.h>

LONG    lElement;    /* current element pointer */
HPS     hps;         /* Presentation-space handle */

/* set the draw mode to retain and open the segment */
if (GpiSetDrawingMode(hps, DM_RETAIN) == TRUE &&
    GpiOpenSegment(hps, 1L) == TRUE)
{
    lElement = GpiQueryElementPointer(hps);
    GpiCloseSegment(hps); /* close the segment */
}
```

GpiQueryElementType – Query Element Type

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryElementType (HPS hps, PLONG piType, LONG iLength, PSZ pszData)

This function returns information about the element to which the element pointer currently points.

Parameters

hps (HPS) – input

Presentation-space handle.

piType (PLONG) – output

Element type.

The element type can be system-defined or application-defined; see GpiElement and GpiBeginElement.

iLength (LONG) – input

Data length.

Length of the description data buffer.

pszData (PSZ) – output

Description of data buffer.

The description may be system-defined or application-defined; see GpiElement and GpiBeginElement. The string is null-terminated, even if it has to be truncated.

Returns

Size of the data required to hold the element content.

This can be used for a subsequent GpiQueryElement function.

≥0 Size of data

GPI_ALTError Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION

An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_NO_CURRENT_ELEMENT

An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open element.

PMERR_NOT_IN_RETAIN_MODE

An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to **retain**

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

PMERR_INV_IN_ELEMENT

An attempt was made to issue a function invalid inside an element bracket.

PMERR_NO_CURRENT_SEG

An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open segment.

GpiQueryElementType – Query Element Type

Remarks

This function is only valid when the drawing mode (see GpiSetDrawingMode) is set to **retain** (not **draw-and-retain**), and a segment bracket is currently in progress. It is not valid in an element bracket.

Example Code

This example uses the GpiQueryElementType function to retrieve the size of the current element. The size is used to retrieve the graphics-order data in the element.

```
#define INCL_GPISEGEDITING      /* GPI Segment Edit functions */
#include <os2.h>

HPS hps;          /* presentation space handle */
BYTE abElement[512];
LONG cbElement;
LONG lType;

/* move pointer to first element in segment */

GpiSetElementPointer(hps, lL);
cbElement = GpiQueryElementType(
    hps,          /* presentation space */
    &lType,        /* variable to receive type */
    0L,          /* copy zero bytes of description */
    NULL);       /* no buffer for description */

GpiQueryElement(hps, 0L, cbElement, abElement);
```


GpiQueryFaceString –

Query Face String

```
#define INCL_GPILCIDS /* Or use INCL_GPI or INCL_PM */
```

```
ULONG GpiQueryFaceString (HPS hps, PSZ pszFamilyName,  
                          PFACENAMEDESC pfindFaceAttrs, LONG lLength,  
                          PSZ pszCompoundFaceName)
```

This function generates a compound face name for a font.

Parameters

hps (HPS) – input

Presentation-space handle.

pszFamilyName (PSZ) – input

Family name.

The family name of the font (for example, "Courier").

pfindFaceAttrs (PFACENAMEDESC) – input

Face-name description.

A structure that provides the characteristics of the required font. These characteristics are used to generate the compound face name.

lLength (LONG) – input

Length of *pszCompoundFaceName* buffer.

The maximum length of the compound face name returned (including the trailing zero of the string).

Specify zero to find out how large the *pszCompoundFaceName* buffer needs to be.

pszCompoundFaceName (PSZ) – output

Compound face name.

The compound face name of the font.

Returns

Length of the compound face name:

GPI_ERROR Error occurred

> 0 Length of the compound face-name string (including the trailing zero). This is the length of the complete string; if it is greater than *lLength*, the string returned in *pszCompoundFaceName* is truncated.

Possible returns from WinGetLastError

PMERR_FONT_NOT_LOADED

An attempt was made to create a font that was not loaded.

PMERR_INV_FACENAME

An invalid font family name was passed to GpiQueryFaceString.

PMERR_INV_FACENAMEDESC

The font facename description is invalid.

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

GpiQueryFaceString — Query Face String

Remarks

This function generates a compound face name (for example, "Courier Bold Italic") from a family name (for example, "Courier").

The compound face name can be used on a GpiCreateLogFont function.

Example Code

This example uses GpiQueryFaceString to generates a compound face name of 'Courier Light Italic' from the family name 'Courier.'

```
#define INCL_GPILCIDS          /* Font functions          */
#include <os2.h>

ULONG  cbRetLength;          /* length of compound face name */
HPS    hps;                  /* Presentation-space handle    */
char    pszFamilyName[13];   /* Family name                   */
FACENAMEDESC  pfndFaceAttrs; /* Face-name description        */
LONG    lLength;             /* length of buffer              */
char    *pszCompoundFaceName; /* Compound face name           */

/* family name is 'Courier' */
strcpy(pszFamilyName,"Courier");

/* let the function determine the buffer length and return it */
lLength = 0L;

/* initialize face name description structure for Light weight
   class, normal width, and italics */
pfndFaceAttrs.usSize = sizeof(FACENAMEDESC);
/* Length of structure */
pfndFaceAttrs.usWeightClass = FWEIGHT_LIGHT; /* Weight class */
pfndFaceAttrs.usWidthClass = FWIDTH_NORMAL; /* Width class */
pfndFaceAttrs.usReserved = 0; /* Reserved */
pfndFaceAttrs.flOptions = FTYPE_ITALIC;
/* Other characteristics of the font */

cbRetLength = GpiQueryFaceString(hps, pszFamilyName,
                                &pfndFaceAttrs, lLength,
                                pszCompoundFaceName);
```

GpiQueryFontAction – Query Font Action

```
#define INCL_GPILCIDS /* Or use INCL_GPI or INCL_PM */
```

ULONG GpiQueryFontAction (HAB hab, ULONG flOptions)

This function determines whether available fonts have been affected since the last time the function was called.

Parameters

hab (HAB) – input
Anchor-block handle.

flOptions (ULONG) – input
Options

The following may be OR'ed together if required:

QFA_PUBLIC Interested in any change of public fonts.

QFA_PRIVATE Interested in any change of private fonts for the current process.

Returns

Actions indicator:

If no error occurs, QFA_PUBLIC and QFA_PRIVATE may be OR'ed together.

QFA_PUBLIC A change of public fonts has occurred.

QFA_PRIVATE A change of private fonts affecting the current process has occurred.

QFA_ERROR Error occurred.

Possible returns from WinGetLastError

PMERR_INV_OR_INCOMPAT_OPTIONS An invalid or incompatible (with micro presentation space) options parameter was specified with GpiCreatePS or GpiSetPS.

Remarks

This function can be used by a font selection dialog to find out whether its database of font information is still valid.

The function returns that both public and private font changes have taken place the first time it is called on a given process.

GpiQueryFontFileDescriptions – Query Font File Descriptions

```
#define INCL_GPILCIDS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryFontFileDescriptions (HAB hab, PSZ pszFilename, PLONG piCount, PFFDESCS affdescsNames)

This function determines whether a given file is a font resource file, and if so, returns the family and face names of the fonts that it contains.

Parameters

hab (HAB) – input

Anchor-block handle.

pszFilename (PSZ) – input

Fully qualified filename.

This is the name of the font resource. The filename extension is .FON.

piCount (PLONG) – input/output

Maximum number of family and face name pairs to be returned.

The number of pairs of descriptions that are actually returned in *affdescsNames* is returned in this variable.

affdescsNames (PFFDESCS) – output

Array of font file descriptors.

An array of $2 \times piCount$ consecutive 32-byte fields, in which the family and face names of each font, in turn, are returned alternately. For each pair, the family name is returned first.

Returns

Returns:

≥ 0 Number of fonts for which details were not returned

GPI_ALTEERROR Error.

Possible returns from WinGetLastError

PMERR_INV_FONT_FILE_DATA

The font file specified with GpiLoadFonts, GpiLoadPublicFonts,

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

Remarks

Details are returned for as many fonts as can be held in *affdescsNames*.

By inspecting the returned data, the application can tell whether a particular font resource file contains the fonts it requires, before loading it.

By specifying *piCount* as 0, and then looking at the value returned in *IRemFonts*, an application can determine how many fonts there are in the file, and then allocate the correct amount of buffer space for a subsequent call to obtain all of the names.

GpiQueryFontFileDescriptions —

Query Font File Descriptions

Example Code

This example uses the GpiQueryFontFileDescriptions to retrieve the typeface family and names for the fonts in the helv.dll file. The function is called twice, once to determine the actual number of fonts in the file, and again to retrieve the descriptions.

```
#define INCL_GPILCIDS          /* Font functions          */
#define INCL_DOSMEMMGR        /* DOS Memory Manager Functions */
#include <os2.h>

HPS hps;          /* presentation space handle */
HAB hab;          /* anchor-block handle       */
PFFDESCS pffdescs; /* array of font file descriptors */
LONG cFonts = 0;  /* number of descriptions not returned */

/* Retrieve a count of all fonts in the file. */

cFonts = GpiQueryFontFileDescriptions(hab, "helv", &cFonts, NULL);

/* Allocate space for the descriptions. */

DosAllocMem((VOID *)pffdescs, (ULONG)(cFonts*sizeof(PFFDESCS)),
            PAG_COMMIT | PAG_READ | PAG_WRITE);

/* Retrieve the descriptions. */

GpiQueryFontFileDescriptions(hab, "helv", &cFonts, pffdescs);
```

```
#define INCL_GPILCIDS /* Or use INCL_GPI or INCL_PM */
```

```
BOOL GpiQueryFontMetrics (HPS hps, LONG IMetricsLength, PFONTMETRICS pfmMetrics)
```

This function returns a record providing details of the font metrics for the logical font that is currently selected.

Parameters

hps (**HPS**) – input
Presentation-space handle.

IMetricsLength (**LONG**) – input
Length of metrics.

pfmMetrics (**PFONTMETRICS**) – output
Metrics of font.

In this buffer are returned the font metrics of the logical font, identified by the current value of the character set attribute.

No more data than *IMetricsLength* is returned.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT An invalid length or count parameter was specified.

PMERR_COORDINATE_OVERFLOW An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

Remarks

All sizes are returned in world coordinates.

An application can determine if the font *szFacename*[*FACESIZE*] (as returned in *pfmMetrics*) has been truncated by checking the *usType* field in *pfmMetrics* for the *FM_TYPE_FACETRUNC* indicator. If the face name has been truncated, this bit will be set, and the application can issue a *WinQueryAtomName* function, passing in the *FaceNameAtom* (as returned in *pfmMetrics*) to retrieve the full face name from the System Atom table.

GpiQueryFontMetrics — Query Font Metrics

Example Code

This example uses the GpiQueryFontMetrics function to retrieve the font metrics for the current font.

```
#define INCL_GPILCIDS          /* Font functions          */
#include <os2.h>

HPS hps;                      /* presentation space handle */
FONTMETRICS fm;               /* metrics structure          */

GpiQueryFontMetrics(hps, sizeof(FONTMETRICS), &fm);
```

GpiQueryFonts – Query Fonts

```
#define INCL_GPILCIDS /* Or use INCL_GPI or INCL_PM */
```

```
LONG GpiQueryFonts (HPS hps, ULONG flOptions, PSZ pszFacename, PLONG plReqFonts,  
LONG lMetricsLength, PFONTMETRICS afmMetrics)
```

This function returns a record providing details of the fonts that match the specified *pszFacename*.

Parameters

hps (HPS) – input
Presentation-space handle.

flOptions (ULONG) – input
Enumeration options.

This controls which fonts are to be enumerated. If both the following options are required, the values should be ORed together:

QF_PUBLIC Enumerate public fonts.

QF_PRIVATE Enumerate private fonts.

QF_NO_DEVICE Device fonts are not reported.

QF_NO_GENERIC Generic fonts are not reported.

pszFacename (PSZ) – input
Face name of fonts.

If the pointer to *pszFacename* is NULL, all available fonts are queried, regardless of their face names.

plReqFonts (PLONG) – input/output
Count of fonts.

The number of fonts for which the application requires the metrics. This variable returns the number of fonts returned.

lMetricsLength (LONG) – input
Length of metrics.

The length of each metrics record to be returned. The *afmMetrics* data area must be *plReqFonts* multiplied by *lMetricsLength* long.

afmMetrics (PFONTMETRICS) – output
Metrics of font.

In this structure are returned the font metrics of up to *plReqFonts* matching fonts. The format for each record is as defined for *GpiQueryFontMetrics*, except that the *usCodePage* field has no meaning in this context, and is indeterminate. For each font, no more data than *lMetricsLength* is returned.

Returns

Count of fonts not returned:

≥ 0 Count of fonts not returned

GPI_ALTEERROR Error.

Possible returns from *WinGetLastError*

PMERR_INV_HPS

An invalid presentation-space handle was specified.

GpiQueryFonts — Query Fonts

PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_COORDINATE_OVERFLOW	An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

Remarks

Font metrics are returned for as many matching fonts as can be held in *afmMetrics*.

By inspecting the returned data, the application can choose which of the available fonts is most appropriate for its requirements. If necessary, it can force selection of a particular font, by specifying its **match** (as returned in *afmMetrics*) in the *pAttrs* structure for *GpiCreateLogFont*, however, this is only valid for a particular device/device driver combination on a single machine. This method should be avoided as a method for selecting a font.

An application can determine if the font *szFacename*[*FACESIZE*] (as returned in *afmMetrics*) has been truncated by checking the *usType* field in *afmMetrics* for the *FM_TYPE_FACETRUNC* indicator. If the face name has been truncated, this bit will be set, and the application can issue a *WinQueryAtomName* function, passing in the *FaceNameAtom* (as returned in *afmMetrics*) to retrieve the full face name from the System Atom table.

By specifying *plReqFonts* as 0, and then looking at the value returned in *lRemFonts*, an application can determine how many fonts there are that match the *pszFacename*.

All sizes are returned in world coordinates.

Example Code

This example uses the *GpiQueryFonts* function to retrieve the font metrics for all private fonts having the "Helv" typeface name. The function is called twice, first to determine the number of fonts available, and then again to retrieve the font metrics for all the fonts.

```
#define INCL_GPILCIDS           /* Font functions */
#define INCL_DOSMEMMGR         /* DOS Memory Manager Functions */
#include <os2.h>

HPS hps;           /* presentation space handle */
LONG cFonts;       /* fonts not returned */
LONG lTemp = 0L;   /* font count */
PFONTMETRICS pfm;  /* metrics structure */

/* Determine the number of fonts. */

cFonts = GpiQueryFonts(hps, QF_PRIVATE, "Helv", &lTemp,
    (LONG) sizeof(FONTMETRICS), NULL);

/* Allocate space for the font metrics. */

DosAllocMem((VOID *) pfm, (ULONG) (cFonts * sizeof(FONTMETRICS)),
    PAG_COMMIT | PAG_READ | PAG_WRITE);

/* Retrieve the font metrics. */

cFonts = GpiQueryFonts(hps, QF_PRIVATE, "Helv", &cFonts,
    (LONG) sizeof(FONTMETRICS), pfm);
```

GpiQueryFullFontFileDescriptions – Query Full Font File Descriptions

```
#define INCL_GPILCIDS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryFullFontFileDescriptions (HAB hab, PSZ pszFilename, PLONG piCount, PVOID pNames, PLONG piNamesBuffLength)

This function determines whether a given file is a font resource file, and if so, returns the family and face names of the fonts that it contains.

Parameters

hab (HAB) – input

Anchor-block handle.

pszFilename (PSZ) – input

Fully qualified filename.

This is the name of the font resource.

piCount (PLONG) – input/output

Maximum number of family and face name pairs to be returned.

The number of pairs of descriptions that are actually returned in *pNames* is returned in this variable.

pNames (PVOID) – output

Font file descriptors.

A buffer in which the font file family and face name pairs are returned. They are each returned in a FFDESCS2 structure, with successive structures packed end to end.

piNamesBuffLength (PLONG) – input/output

Length, in bytes, of the *pNames* data buffer.

On return, this is set to the actual length needed to hold all of the family names and face names in the file.

Returns

Returns:

≥0 Number of fonts for which details were not returned

GPI_ALTERROR Error.

Possible returns from WinGetLastError

PMERR_INV_FONT_FILE_DATA

The font file specified with GpiLoadFonts, GpiLoadPublicFonts,

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

Remarks

Details are returned for as many fonts as can be held in *pNames*.

By inspecting the returned data, the application can tell whether a particular font resource file contains the fonts it requires, before loading it.

By specifying *pNames* as **NULL**, and then looking at the value returned in *piNamesBuffLength*, an application can determine the length of the buffer needed to hold all of the font names.

Support for this function is device dependent.

GpiQueryFullFontFileDescriptions — Query Full Font File Descriptions

Example Code

```
/* This example uses the GpiQueryFullFontFileDescriptions to */
/* retrieve the typeface family and names for the fonts in the */
/* helv.dll file. The function is called twice, once to */
/* determine the actual number of fonts in the file, and again */
/* to retrieve the descriptions. */

PFFDESCS pffdescs;
SEL sel;
LONG cFonts = 0;
LONG lBuflen = 0;

/* Retrieve a count of all fonts in the file. */

cFonts = GpiQueryFontFileDescriptions(hab, "helv",
    &cFonts, NULL, &lBuflen)

/* Allocate space for the descriptions. */

DosAllocSeg((USHORT) lBuflen, &sel, SEG_NONSHARED);
pffdescs = MAKEP(sel, 0);

/* Retrieve the descriptions. */

GpiQueryFullFontFileDescriptions(hab, "helv", &cFonts,
    pffdescs, &lBuflen);
```

GpiQueryGraphicsField – Query Graphics Field

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryGraphicsField (HPS hps, PRECTL prclField)

This function returns the bottom-left and top-right corners of the graphics field in presentation page units, as set by the GpiSetGraphicsField function.

Parameters

hps (HPS) – input
Presentation-space handle.

prclField (PRECTL) – output
Graphics field.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.

Example Code

This example uses GpiQueryGraphicsField to return the bottom-left and top-right corners of the graphics field in presentation page units, as set by the GpiSetGraphicsField call, and then assigns the x coordinate of the lower left hand field corner to a variable.

```
#define INCL_GPITRANSFORMS /* Transform functions */
#include <os2.h>

BOOL fSuccess; /* success indicator */
HPS hps; /* Presentation-space handle */
RECTL prclField; /* graphics field */
LONG lLwrLftxCoord; /* lower left x coordinate of field */

fSuccess = GpiQueryGraphicsField(hps, &prclField);

/* if successful, assign lower left x coordinate of graphics field */
if (fSuccess == TRUE)
    lLwrLftxCoord = prclField.xLeft;
```

GpiQueryInitialSegmentAttrs – Query Initial Segment Attributes

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryInitialSegmentAttrs (HPS hps, LONG IAttribute)

This function returns the current value of a particular initial segment attribute.

Parameters

hps (HPS) – input
Presentation-space handle.

IAttribute (LONG) – input
Attribute to be queried.

Identifies the attribute to be returned by this function:

ATTR_DETECTABLE	Detectability
ATTR_VISIBLE	Visibility
ATTR_CHAINED	Chained
ATTR_DYNAMIC	Dynamic
ATTR_FASTCHAIN	Fast chaining
ATTR_PROP_DETECTABLE	Propagate detectability
ATTR_PROP_VISIBLE	Propagate visibility.

Returns

Current initial attribute value:

ATTR_ON	On/yes
ATTR_OFF	Off/no
ATTR_ERROR	Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_SEG_ATTR	An invalid attribute parameter was specified with GpiSetSegmentAttrs, GpiQuerySegmentAttrs, GpiSetInitialSegmentAttrs, or GpiQueryInitialSegmentAttrs.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.

Remarks

Initial segment attributes are modal settings used to determine the initial attributes of new segments as those new segments are created; see GpiSetInitialSegmentAttrs.

GpiQueryInitialSegmentAttrs – Query Initial Segment Attributes

Example Code

This example uses GpiQueryInitialSegmentAttrs to queries the current state of the dynamic segment attribute.

```
#define INCL_GPISEGMENTS      /* Segment functions          */
#include <os2.h>

LONG    lValue;              /* current element pointer    */
HPS     hps;                 /* Presentation-space handle   */
LONG     lAttribute;         /* attribute to query          */

/* query the dynamic attribute */
lAttribute = ATTR_DYNAMIC;

lValue = GpiQueryInitialSegmentAttrs(hps, lAttribute);
```

GpiQueryKerningPairs —

Query Kerning Pairs

```
#define INCL_GPILCIDS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryKerningPairs (HPS hps, LONG ICount, PKERNINGPAIRS akrnprData)

This function returns kerning pair information for the logical font identified by the current value of the character set attribute.

Parameters

- hps** (HPS) – input
Presentation-space handle.
- ICount** (LONG) – input
Number of elements in *akrnprData*.
- akrnprData** (PKERNINGPAIRS) – output
Kerning pairs.

An array of *ICount* kerning pairs in which information is returned. No more than *ICount* records are returned.

Returns

- Number returned and error indicators:
- ≥ 0 Number of kerning pairs returned
- GPI_ALTEERROR** Error.

Possible returns from WinGetLastError

- | | |
|----------------------------------|---|
| PMERR_INV_HPS | An invalid presentation-space handle was specified. |
| PMERR_PS_BUSY | An attempt was made to access the presentation space from more than one thread simultaneously. |
| PMERR_INV_LENGTH_OR_COUNT | An invalid length or count parameter was specified. |
| PMERR_COORDINATE_OVERFLOW | An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large. |

Remarks

The number of kerned pairs is a field in the font metrics.

GpiQueryKerningPairs – Query Kerning Pairs

Example Code

This example uses the GpiQueryKerningPairs function to retrieve the kerning information for the current font.

```
#define INCL_GPILCIDS          /* Font functions          */
#define INCL_DOSMEMMGR        /* DOS Memory Manager Functions */
#include <os2.h>

HPS hps;          /* presentation space handle */
FONTMETRICS fm;   /* metrics structure          */
PKERNINGPAIRS akrnpr; /* kerning pairs array      */

GpiQueryFontMetrics(hps, (LONG) sizeof(FONTMETRICS), &fm);

DosAllocMem((VOID *)akrnpr,
            (ULONG)(fm.sKerningPairs * sizeof(KERNINGPAIRS)),
            PAG_COMMIT | PAG_READ | PAG_WRITE);

GpiQueryKerningPairs(hps, (LONG) fm.sKerningPairs, akrnpr);
```


GpiQueryLineEnd — Query Line End

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryLineEnd (HPS hps)

This function returns the current line-end attribute.

Parameters

hps (HPS) — input
Presentation-space handle.

Returns

Line end:

LINEEND_DEFAULT Default

>0 Line end

LINEEND_ERROR Error.

Possible returns from GetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Example Code

This example uses GpiQueryLineEnd to return the current line-end attribute after setting the draw mode to DRAW.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions      */
#define INCL_GPICONTROL        /* Control functions    */
#include <os2.h>

HPS    hps;                    /* Presentation-space handle */
LONG   lLineEnd;               /* Line end                */

if (GpiSetDrawingMode(hps, DM_DRAW) == TRUE)
    lLineEnd = GpiQueryLineEnd(hps);
```

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

```
LONG GpiQueryLineJoin (HPS hps)
```

This function returns the current line-join attribute, as set by the GpiSetLineJoin function.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Line join:

LINEJOIN_DEFAULT	Default
>0	Line join
LINEJOIN_ERROR	Error.

Possible returns from GetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Example Code

This example uses GpiQueryLineJoin to return the current line-join attribute after setting the draw mode to DRAW.

```
#define INCL_GPIPRIMITIVES    /* Primitive functions    */
#define INCL_GPICONTROL      /* Control functions    */
#include <os2.h>

HPS    hps;                  /* Presentation-space handle    */
LONG    lLineJoin;           /* Line join                    */

if (GpiSetDrawingMode(hps, DM_DRAW) == TRUE)
    lLineJoin = GpiQueryLineJoin(hps);
```

GpiQueryLineType — Query Line Type

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryLineType (HPS hps)

This function returns the current cosmetic line-type attribute, as set by the GpiSetLineType function.

Parameters

hps (HPS) — input
Presentation-space handle.

Returns

Line type:

LINETYPE_DEFAULT Default
>0 Line type
LINETYPE_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Example Code

This example uses GpiQueryLineType to return the current cosmetic line-type attribute after setting the draw mode to DRAW.

```
#define INCL_GPIPRIMITIVES    /* Primitive functions    */
#define INCL_GPICONTROL      /* Control functions */
#include <os2.h>

HPS    hps;                /* Presentation-space handle */
LONG   lLineType;          /* Line type                 */

if (GpiSetDrawingMode(hps, DM_DRAW) == TRUE)
    lLineType = GpiQueryLineType(hps);
```

GpiQueryLineWidth – Query Line Width

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

FIXED GpiQueryLineWidth (HPS hps)

This function returns the current value of the cosmetic line-width attribute, as set by the GpiSetLineWidth function.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Line width:

LINEWIDTH_DEFAULT	Default
>0	Line width
LINEWIDTH_ERROR	Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Example Code

This example uses GpiQueryLineWidth to return the current cosmetic line-width attribute after setting the draw mode to DRAW.

```
#define INCL_GPIPRIMITIVES /* Primitive functions */
#define INCL_GPICONTROL /* Control functions */
#include <os2.h>

HPS hps; /* Presentation-space handle */
FIXED fxLineWidth; /* Line width */

if (GpiSetDrawingMode(hps, DM_DRAW) == TRUE)
    fxLineWidth = GpiQueryLineWidth(hps);
```

GpiQueryLineWidthGeom —

Query Line Width Geom

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryLineWidthGeom (HPS hps)

This function returns the current geometric line-width attribute, as set by the GpiSetLineWidthGeom function.

Parameters

hps (HPS) — input
Presentation-space handle.

Returns

Geometric line width:

If the geometric line width is currently set to the default, zero is returned.

≥ 0 Geometric line width

LINEWIDTHGEOM_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Example Code

This example uses GpiQueryLineWidthGeom to return the current geometric line-width attribute after setting the draw mode to DRAW.

```
#define INCL_GPIPRIMITIVES /* Primitive functions */
#define INCL_GPICONTROL /* Control functions */
#include <os2.h>

HPS hps; /* Presentation-space handle */
LONG lLineWidth; /* geometric line width */

if (GpiSetDrawingMode(hps, DM_DRAW) == TRUE)
    lLineWidth = GpiQueryLineWidthGeom(hps);
```

GpiQueryLogColorTable – Query Logical Color Table

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryLogColorTable (HPS hps, ULONG flOptions, LONG IStart, LONG ICount, PLONG alArray)

This function returns the logical color table.

Parameters

hps (HPS) – input
Presentation-space handle.

flOptions (ULONG) – input
Specifies options:

LCOLOPT_INDEX

B'1' The index is to be returned for each RGB value

Other flags are reserved and must be B'0'.

IStart (LONG) – input
Starting index for which data is to be returned. This must be greater than or equal to zero.

ICount (LONG) – input
Count of elements.

Number of elements available in *alArray*.

alArray (PLONG) – output
Array in which the information is returned.

If the LCOLOPT_INDEX flag is B'0', it is an array of RGB values (each value is as defined for GpiCreateLogColorTable), starting with the specified index, and ending either when there are no further loaded entries in the table, or when *alArray* has been exhausted. If the logical color table is not loaded with a contiguous set of indexes, QLCT_NOTLOADED is returned as the RGB value for any index values, outside the default range, that have not been explicitly loaded.

If the LCOLOPT_INDEX flag is B'1', it is an array of alternating color indexes and RGB values, in the order index1, RGB value1, index2, RGB value2,... An even number of elements is always returned. If the logical color table is not loaded with a contiguous set of indexes, any index values that are not loaded are skipped.

Returns

Number of elements returned and error indicators:

QLCT_RGB Table in RGB mode, no elements returned

>0 Number of elements returned

QLCT_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

PMERR_INV_COLOR_OPTIONS

An invalid options parameter was specified with a logical color table or color query function.

GpiQueryLogColorTable — Query Logical Color Table

PMERR_INV_COLOR_START_INDEX	An invalid starting index parameter was specified with a logical color table or color query function.
PMERR_PALETTE_SELECTED	Color palette operations cannot be performed on a presentation space while a palette is selected.

Example Code

This example uses the GpiQueryLogColorTable function to retrieve all the entries in the current logical color table.

```
#define INCL_GPILOGCOLORTABLE /* Color Table functions */
#define INCL_DOSMEMMGR      /* DOS Memory Manager Functions */
#define INCL_DEV            /* Device Function definitions */
#include <os2.h>

HPS hps; /* presentation space handle */
LONG cColors; /* number of colors */
PLONG aIColor; /* color table array */

/* Find out how many colors are in the color table. */
DevQueryCaps(GpiQueryDevice(hps), CAPS_COLORS, 1L, &cColors);

/* Allocate space for the color values and indexes. */
DosAllocMem((VOID *)aIColor, (ULONG)cColors*2,
            PAG_COMMIT | PAG_READ | PAG_WRITE);

/* Retrieve the values. */
GpiQueryLogColorTable(hps, /* presentation space */
                     LCOLOPT_INDEX, /* retrieve indexes and RGB values */
                     0L, /* start with first entry */
                     cColors * 2, /* copy 2 values for each entry */
                     aIColor); /* array to receive values */
```

GpiQueryLogicalFont – Query Logical Font

```
#define INCL_GPILCIDS /* Or use INCL_GPI or INCL_PM */
```

```
BOOL GpiQueryLogicalFont (HPS hps, LONG lCid, PSTR8 pName, PFATTRS pfatAttrs,  
                           LONG lAttrsLength)
```

This function returns the description of a logical font. See GpiCreateLogFont.

Parameters

- hps** (HPS) – input
Presentation-space handle.
- lCid** (LONG) – input
Local identifier.
Logical font local identifier, in the range 0 through 254. Specify 0 to query the default font.
- pName** (PSTR8) – output
Logical font name.
An 8-character name for the logical font.
- pfatAttrs** (PFATTRS) – output
Attributes of font.
- lAttrsLength** (LONG) – input
Length of *pfatAttrs* buffer.
The maximum length of font attribute data to be returned.

Returns

- Success indicator:
- TRUE** Successful completion
- FALSE** Error occurred.

Possible returns from WinGetLastError

- | | |
|----------------------------------|--|
| PMERR_INV_HPS | An invalid presentation-space handle was specified. |
| PMERR_PS_BUSY | An attempt was made to access the presentation space from more than one thread simultaneously. |
| PMERR_INV_SETID | An invalid setid parameter was specified. |
| PMERR_SETID_IN_USE | An attempt was made to specify a setid that was already in use as the currently selected character, marker or pattern set. |
| PMERR_INV_LENGTH_OR_COUNT | An invalid length or count parameter was specified. |

Remarks

If the specified local identifier is in use to tag a bit map (see GpiSetBitmapId), an error is raised.

GpiQueryLogicalFont — Query Logical Font

Example Code

This example uses GpiQueryLogicalFont to return the description of the default logical font and if the query succeeds, assigns the font code page to a variable.

```
#define INCL_GPILCIDS          /* Font functions          */
#include <os2.h>

BOOL fSuccess;                /* success indicator    */
HPS hps;                      /* Presentation-space handle */
LONG lLcid;                   /* local identifier      */
PSTR8 pName;                  /* 8 character logical font name */
PFATTRS pfatAttrs;            /* Attributes of font    */
LONG lAttrsLength;            /* length of buffer      */
USHORT usFontCodePage;        /* font code page        */

/* query the default font */
lLcid = 0L;

/* return all information */
lAttrsLength = sizeof(FATTRS);

fSuccess = GpiQueryLogicalFont(hps, lLcid, pName, pfatAttrs,
                               lAttrsLength);

/* if successful, assign value of font code page */
if (fSuccess == TRUE)
    usFontCodePage = pfatAttrs->usCodePage;
```

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryMarker (HPS hps)

This function returns the current value of the marker symbol attribute, as set by the GpiSetMarker function.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Marker symbol:

MARKSYM_DEFAULT	Default
>0	Marker symbol
MARKSYM_ERROR	Error.

Possible returns from GetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Example Code

This example uses GpiQueryMarker to return the current marker symbol attribute after setting the draw mode to DRAW.

```
#define INCL_GPIPRIMITIVES /* Primitive functions */
#define INCL_GPICONTROL /* Control functions */
#include <os2.h>

HPS hps; /* Presentation-space handle */
LONG lSymbol; /* marker symbol */

if (GpiSetDrawingMode(hps, DM_DRAW) == TRUE)
    lSymbol = GpiQueryMarker(hps);
```

GpiQueryMarkerBox – Query Marker Box

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryMarkerBox (HPS hps, PSIZEF pszfxSize)

This function returns the current value of the marker-box attribute, as set by the GpiSetMarkerBox function.

Parameters

hps (HPS) – input
Presentation-space handle.

pszfxSize (PSIZEF) – output
Size of marker box.

The size of the marker box is in world coordinates.

If the marker box is currently set to the default, the default size is returned.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryMarkerBox – Query Marker Box

Example Code

This example uses GpiQueryMarkerBox to return the current marker-box attribute after setting the draw mode to DRAW.

```
#define INCL_GPIPRIMITIVES    /* Primitive functions    */
#define INCL_GPICONTROL      /* Control functions    */
#include <os2.h>

BOOL fSuccess;               /* success indicator    */
HPS  hps;                    /* Presentation-space handle */
SIZEF psizfxSize;            /* size of marker-box    */
FIXED lWidth;                /* marker-box width      */

if (GpiSetDrawingMode(hps, DM_DRAW) == TRUE)
    fSuccess = GpiQueryMarkerBox(hps, &psizfxSize);

/* if successful, assign value of marker-box width */
if (fSuccess == TRUE)
    lWidth = psizfxSize.cx;
```

GpiQueryMarkerSet — Query Marker Set

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

```
LONG GpiQueryMarkerSet (HPS hps)
```

This function returns the current value of the marker-set attribute, as set by the GpiSetMarkerSet function.

Parameters

hps (HPS) — input
Presentation-space handle.

Returns

Marker-set local identifier:

LCID_DEFAULT Default
>0 Marker-set local identifier
LCID_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Example Code

This example uses GpiQueryMarkerSet to return the current marker-set attribute after setting the draw mode to DRAW.

```
#define INCL_GPIPRIMITIVES /* Primitive functions */
#define INCL_GPICONTROL /* Control functions */
#include <os2.h>

HPS hps; /* Presentation-space handle */
LONG lSet; /* marker-set local identifier */

if (GpiSetDrawingMode(hps, DM_DRAW) == TRUE)
    lSet = GpiQueryMarkerSet(hps);
```

GpiQueryMetaFileBits – Query Metafile Bits

```
#define INCL_GPIMETAFILES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryMetaFileBits (HMF hmf, LONG IOffset, LONG ILength, PBYTE pbData)

This function transfers a metafile to application storage.

Parameters

hmf (HMF) – input
Memory-metafile handle.

IOffset (LONG) – input
Byte offset.

Offset into the metafile data from which the transfer must start. This is useful in instances where the metafile data is too long to fit into a single application buffer.

ILength (LONG) – input
Length in bytes of the metafile data to copy.

pbData (PBYTE) – output
Metafile data.

Address in application storage into which the metafile data is copied.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HMF An invalid metafile handle was specified.

PMERR_INV_METAFILE_LENGTH An invalid length parameter was specified with GpiSetMetaFileBits or GpiQueryMetaFileBits.

PMERR_INV_METAFILE_OFFSET An invalid length parameter was specified with GpiSetMetaFileBits or GpiQueryMetaFileBits.

PMERR_METAFILE_IN_USE An attempt has been made to access a metafile that is in use by another thread.

PMERR_TOO_MANY_METAFILES_IN_USE The maximum number of metafiles allowed for a given process was exceeded.

Remarks

The total length of a metafile can be found from the data returned by GpiQueryMetaFileLength. This function allows an application to retrieve the data in units of a manageable size.

GpiQueryMetaFileBits —

Query Metafile Bits

Example Code

This example uses the GpiQueryMetaFileBits function to retrieve the graphics-order data from the specified metafile. The GpiQueryMetaFileLength function returns the length of the metafile.

```
#define INCL_GPIMETAFILES      /* Metafile functions */
#define INCL_DOSMEMMGR        /* DOS Memory Manager Functions */
#include <os2.h>

HPS hps;          /* presentation space handle */
HMF hmf;          /* metafile handle */
LONG cBytes;      /* metafile length */
LONG off;         /* metafile byte offset */
PBYTE pbBuffer;   /* metafile data buffer */

hmf = GpiLoadMetaFile(hps, "sample.met");

cBytes = GpiQueryMetaFileLength(hmf); /* gets length of metafile */

/* Allocate the buffer for the metafile data. */

DosAllocMem((VOID *)pbBuffer, (ULONG)cBytes,
            PAG_COMMIT | PAG_READ | PAG_WRITE);

GpiQueryMetaFileBits(
    hmf,          /* handle of metafile */
    off,          /* offset of next byte to retrieve */
    cBytes,       /* length of data */
    pbBuffer);    /* buffer to receive metafile data */
```

GpiQueryMetaFileLength – Query Metafile Length

```
#define INCL_GPIMETAFILES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryMetaFileLength (HMF hmf)

This function returns the total length of a memory metafile, in bytes.

Parameters

hmf (HMF) – input
Memory-metafile handle.

Returns

Total length of the metafile:

≥0 Total length of the metafile

GPI_ALTError Error.

Possible returns from WinGetLastError

PMERR_INV_HMF An invalid metafile handle was specified.

PMERR_TOO_MANY_METAFILES_IN_USE The maximum number of metafiles allowed for a given process was exceeded.

Remarks

This function is normally used before GpiQueryMetaFileBits.

Example Code

This example uses GpiQueryMetaFileLength to query the byte length of a memory metafile.

```
#define INCL_GPIMETAFILES /* Meta File functions */
#include <os2.h>

HMF hmf; /* memory-metafile handle */
LONG lLength; /* length of metafile */

lLength = GpiQueryMetaFileLength(hmf);
```


GpiQueryMix —

Query Mix

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryMix (HPS hps)

This function returns the current value of the (character) foreground color-mixing mode, as set by the GpiSetMix function.

Parameters

hps (HPS) — input
Presentation-space handle.

Returns

Mix mode:

FM_DEFAULT	Default
>0	Mix mode
FM_ERROR	Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Example Code

This example uses GpiQueryMix to return the current foreground-mixing mode after setting the draw mode to DRAW.

```
#define INCL_GPIPRIMITIVES /* Primitive functions */
#define INCL_GPICONTROL /* Control functions */
#include <os2.h>

HPS hps; /* Presentation-space handle */
LONG lMixMode; /* mix mode */

if (GpiSetDrawingMode(hps, DM_DRAW) == TRUE)
    lMixMode = GpiQueryMix(hps);
```

GpiQueryModelTransformMatrix – Query Model Transform Matrix

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryModelTransformMatrix (HPS hps, LONG ICount, PMATRIXLF pmatlfArray)

This function returns the current model transform; see GpiSetModelTransformMatrix.

Parameters

hps (HPS) – input
Presentation-space handle.

ICount (LONG) – input
Number of elements.

The number of elements to be returned in *pmatlfArray*; must be in the range 0 through 9. If 0 is specified, no matrix elements are returned.

pmatlfArray (PMATRIXLF) – output
Transform matrix.

A structure in which the elements of the model transform matrix are returned.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from GetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE

An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

PMERR_INV_DC_TYPE

An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryModelTransformMatrix — Query Model Transform Matrix

Example Code

This example uses GpiQueryModelTransformMatrix to query the first element of the current model transform after setting the draw mode to DRAW.

```
#define INCL_GPITRANSFORMS    /* Transform functions      */
#define INCL_GPICONTROL      /* Control functions  */
#include <os2.h>

BOOL fSuccess;              /* success indicator   */
HPS  hps;                   /* Presentation-space handle */
LONG lCount;                /* number of elements  */
MATRIXLF pmatlfArray;       /* transform matrix     */

lCount = 1; /* examine only first element of transform matrix */

if (GpiSetDrawingMode(hps, DM_DRAW) == TRUE)
    fSuccess = GpiQueryModelTransformMatrix(hps, lCount,
                                             &pmatlfArray);
```

GpiQueryNearestColor – Query Nearest Color

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryNearestColor (HPS hps, ULONG ulOptions, LONG IRgbIn)

This function returns the nearest color available to the color specified on the currently associated device. Both colors are specified in RGB terms.

Parameters

hps (HPS) – input
Presentation-space handle.

ulOptions (ULONG) – input
Options:

Reserved, must be zero.

IRgbIn (LONG) – input
Required color.

Returns

Nearest available color to the one specified:

≥0 Nearest available color

GPI_ALTError Error.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COLOR_OPTIONS An invalid options parameter was specified with a logical color table or color query function.

PMERR_INV_RGBCOLOR An invalid rgb color parameter was specified with GpiQueryNearestColor or GpiQueryColor

Remarks

The nearest color returned is one that is available in the *physical* palette on the device. This might not actually be available with the currently loaded logical color table.

The color returned is a pure color, that is, one that can be used for drawing lines, text, and so on. It does not take into account the possibility of *dithered* colors being used for filled areas. With *dithering*, it is likely that the color used for filling areas is different from that used for lines, and text, when the same color index is selected.

For a monochrome device, if *IRgbIn* is the reset color, then *IRgbOut* is also the reset color; otherwise, it is black if the reset color is white, and the converse.

GpiQueryNearestColor — Query Nearest Color

Example Code

This example uses GpiQueryNearestColor to return the nearest color available to the one specified, on the currently associated device.

```
#define INCL_GPILOGCOLORTABLE  /* Color Table functions      */
#include <os2.h>

LONG  lRgbOut;      /* nearest color      */
HPS   hps;          /* Presentation-space handle */
ULONG ulOptions;    /* options            */
LONG  lRgbIn;       /* color to match     */

/* reserved; set to 0 */
ulOptions = 0L;

/* color to find index for */
lRgbIn = (PC_RESERVED*16777216) + (0*65536) + (0*256) + 1;

lRgbOut = GpiQueryNearestColor(hps, ulOptions, lRgbIn);
```

GpiQueryNumberSetIds – Query Number Set Identifiers

```
#define INCL_GPILCIDS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryNumberSetIds (HPS hps)

This function returns the number of local identifiers (lcids) currently in use, referring to fonts or bit maps.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Number of lcids:

≥0 Number of lcids in use

GPI_ALTError Error.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

Remarks

LCID_DEFAULT is included if the default font has been changed (see GpiCreateLogFont).

The information returned by this call can be used to perform a subsequent GpiQuerySetIds request.

Example Code

This example uses GpiQueryNumberSetIds to return the number of local identifiers in use (font and bit map).

```
#define INCL_GPILCIDS                      /* Font functions                      */
#include <os2.h>

LONG    lCount;                      /* number of lcid's                      */
HPS    hps;                      /* Presentation-space handle              */

lCount = GpiQueryNumberSetIds(hps);
```

GpiQueryPageViewport — Query Page Viewport

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryPageViewport (HPS hps, PRECTL prclViewport)

This function returns the page viewport; see GpiSetPageViewport.

Parameters

hps (HPS) — input
Presentation-space handle.

prclViewport (PRECTL) — output
Page viewport.

The size and position of the page viewport in device units.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function must not be issued when there is no device context associated with the presentation space.

GpiQueryPageViewport — Query Page Viewport

Example Code

This example uses GpiQueryPageViewport to query the page viewport, after associating a device context to the presentation space; if successful, it assigns the x coordinate of the viewport to a variable.

```
#define INCL_GPITRANSFORMS      /* Transform functions      */
#include <os2.h>

BOOL fSuccess;                /* success indicator      */
HPS hps;                      /* Presentation-space handle */
RECTL prclViewport;          /* page.viewport          */
LONG lLwrLftxCoord;          /* lower left x coordinate of field */
HDC hdc;                     /* device context handle   */

/* associate device context */
if (GpiAssociate(hps, hdc) == TRUE)
{
    fSuccess = GpiQueryPageViewport(hps, &prclViewport);

    /* if successful, assign lower left x coordinate of viewport */
    if (fSuccess == TRUE)
        lLwrLftxCoord = prclViewport.xLeft;
}
```


GpiQueryPalette —

Query Palette

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI or INCL_PM */
```

HPAL GpiQueryPalette (HPS hps)

This function returns the handle of the palette currently selected into a presentation space.

Parameters

hps (HPS) — input
Presentation-space handle.

Returns

Palette handle.

NULLHANDLE Null handle (no palette is selected)

PAL_ERROR Error occurred

Otherwise Handle of the palette currently selected into this presentation space.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

Remarks

It is possible for a palette to be selected into more than one presentation space at any one time. See GpiSelectPalette.

Example Code

This example uses GpiQueryPalette to return the handle of the palette currently selected into a presentation space and then calls GpiDeletePalette to delete the palette.

```
#define INCL_GPILOGCOLORTABLE /* Color Table functions */
#include <os2.h>

HPAL hpal; /* palette handle */
HPS hps; /* Presentation-space handle */
BOOL fSuccess; /* success indicator */

/* get handle of currently associated palette */
hpal = GpiQueryPalette(hps);

/* delete palette */
fSuccess = GpiDeletePalette(hpal);
```

GpiQueryPaletteInfo – Query Palette Info

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI or INCL_PM */
```

**LONG GpiQueryPaletteInfo (HPAL hpal, HPS hps, ULONG flOptions, LONG IStart,
LONG ICount, PLONG alArray)**

This function passes back the information for a palette.

Parameters

hpal (HPAL) – input
Palette handle.

hps (HPS) – input
Presentation-space handle.

flOptions (ULONG) – input
Specifies options:

LCOLOPT_INDEX If this is set, the index is to be returned for each RGB value in the *alArray* parameter.

Other flags are reserved and must be 0.

IStart (LONG) – input
The starting index for which data is to be returned.

ICount (LONG) – input
Count of elements.
Number of elements available in *alArray*.

If 0 is specified, the number of elements required to return the palette information in *alArray* is returned.

alArray (PLONG) – output
An array in which the palette information is returned.

If **LCOLOPT_INDEX** is not specified, this is an array of RGB values (each value is as defined for *GpiCreatePalette*), starting with the specified index, and ending either when there are no further entries in the palette, or when *alArray* has been exhausted. If the palette is not loaded with a contiguous set of indices, **QLCT_NOTLOADED** is returned as the RGB value for any index values, outside the default range, that have not been explicitly loaded.

If **LCOLOPT_INDEX** is specified, this is an array of alternating color indices and RGB values, in the order index1, RGB value1, index2, RGB value2, An even number of elements is always returned. If the palette is not loaded with a contiguous set of indices, any index values that are not present are skipped.

Returns

Number of elements:

PAL_ERROR Error occurred

Otherwise The number of elements of palette information passed back in the *alArray* parameter, unless *ICount* parameter is 0, in which case this is the total number of elements that are needed to hold the palette information.

Zero is returned if no palette is selected.

Possible returns from *WinGetLastError*

PMERR_INV_HPS

An invalid presentation-space handle was specified.

GpiQueryPaletteInfo — Query Palette Info

PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_HPAL	An invalid color palette handle was specified.
PMERR_INV_COLOR_OPTIONS	An invalid options parameter was specified with a logical color table or color query function.
PMERR_INV_COLOR_START_INDEX	An invalid starting index parameter was specified with a logical color table or color query function.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_PALETTE_BUSY	An attempt has been made to reset the owner of a palette when it was busy.

Remarks

The information passed back is in the same format as that required to create a palette (see `GpiCreatePalette`).

If a non-NULL palette handle is passed in the *hpal* parameter, the information is returned for that palette, and the *hps* parameter is ignored. Otherwise, *hps* identifies a presentation space for which the default colors are returned as a palette.

Note: In this case the default colors are returned, even if a logical color table is currently loaded into the presentation space.

Example Code

This example uses `GpiQueryPaletteInfo` to query the palette information and, if any values are returned, assigns the palette's first color value to a variable.

```
#define INCL_GPILOGCOLORTABLE  /* Color Table functions      */
#include <os2.h>

LONG lRetCount;      /* number of elements      */
HPAL hpal;           /* palette handle          */
ULONG flOptions;     /* options                 */
ULONG ulStart;       /* starting index          */
ULONG ulCount;       /* count of elements in array */
ULONG *aulArray;     /* palette information array */
ULONG ulFirstColor;  /* first color in palette  */

/* specify no options */
flOptions = 0L;

/* start at index 0 */
ulStart = 0L;

/* tell function to determine element count */
ulCount = 0L;

lRetCount = GpiQueryPaletteInfo(hpal, NULLHANDLE, flOptions,
                               ulStart, ulCount,
                               aulArray);

/* if palette info returned, assign value of first color */
ulFirstColor = aulArray[0];
```

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

LONG GpiQueryPattern (HPS hps)

This function returns the current value of the shading-pattern symbol, as set by the GpiSetPattern function.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Pattern symbol:

PATSYM_DEFAULT Default

>0 Pattern symbol

PATSYM_ERROR Error.

Possible returns from GetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Example Code

In this example we query the current value of the shading-pattern symbol, as set by the GpiSetPattern call.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

LONG lResult;          /* pattern symbol if > 0 */
HPS hps;               /* Presentation space handle. */
if(PATSYM_SOLID == GpiQueryPattern(hps))
{
    /* . */
    /* . */
}
```

GpiQueryPatternRefPoint – Query Pattern Reference Point

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryPatternRefPoint (HPS hps, PPOINTL pptlRefPoint)

This function returns the current pattern reference point, as set by the GpiSetPatternRefPoint function.

Parameters

hps (HPS) – input
Presentation-space handle.

pptlRefPoint (PPOINTL) – output
Pattern reference point.

If the pattern reference point is currently set to the default, (0,0) is returned.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Example Code

In this example we query the pattern reference point, which is set by the GpiSetPatternRefPoint.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
BOOL flResult;
HPS hps;          /* Presentation space handle. */
POINTL ptlRefPoint; /* pattern reference point */
LONG xcoord, ycoord;
flResult = GpiQueryPatternRefPoint(hps,
                                   &ptlRefPoint );

xcoord = ptlRefPoint.x; ycoord = ptlRefPoint.y;
```

GpiQueryPatternSet – Query Pattern Set

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryPatternSet (HPS hps)

This function returns the current value of the pattern-set identifier, as set by the GpiSetPatternSet function.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Pattern-set local identifier:

LCID_DEFAULT	Default
>0	Pattern set
LCID_ERROR	Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is not valid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Example Code

In this example we query the pattern set identifier, which is set by the GpiSetPatternSet.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

LONG lpatternset;
HPS hps;          /* Presentation space handle. */

lpatternset = GpiQueryPatternSet(hps);
```

GpiQueryPel — Query Pel

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryPel (HPS hps, PPOINTL pptlPoint)

This function returns the color of a pel at a position specified in world coordinates.

Parameters

hps (HPS) — input
Presentation-space handle.

pptlPoint (PPOINTL) — input
Position in world coordinates.

It is an error if the specified point is outside any of the current clipping objects (that is, clip path, viewing limits, clip region, or visible region).

Returns

Color index of the pel:

≥0 Color of the pel

CLR_NOINDEX No valid index (color is not in logical color table)

GPI_ALTEERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE An invalid coordinate value was specified.

PMERR_PEL_IS_CLIPPED An attempt was made to query a pel that had been clipped using GpiQueryPel.

PMERR_PEL_NOT_AVAILABLE An attempt was made to query a pel that did not exist in GpiQueryPel (for example, a memory device context with no selected bit map).

PMERR_NO_BITMAP_SELECTED An attempt has been made to operate on a memory device context that has no bit map selected.

PMERR_INV_DC_TYPE An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

The color returned is a color index or RGB value, according to the logical color table in force (see GpiCreateLogColorTable).

Example Code

In this example we query the color of a pel at a position specified in world coordinates.

```
#define INCL_GPIBITMAPS
#include <OS2.H>

LONG lcolorindex;      /* color index of pel. */
HPS hps;               /* Presentation space handle. */
POINTL ptlPoint;       /* position in world coordinates. */
LONG xcoord, ycoord;
GpiQueryPel(hps, &ptlPoint);
xcoord = ptlPoint.x; ycoord = ptlPoint.y;
```


GpiQueryPickAperturePosition — Query Pick Aperture Position

```
#define INCL_GPICORRELATION /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryPickAperturePosition (HPS hps, PPOINTL pptlPoint)

This function returns the position of the center of the pick aperture.

Parameters

hps (HPS) — input
Presentation-space handle.

pptlPoint (PPOINTL) — output
Pick-aperture position.

Position of the center of the pick aperture, in presentation-page coordinates.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_DC_TYPE

An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Related Functions

- GpiQueryPickAperturePosition
- GpiSetPickAperturePosition
- GpiQueryPickApertureSize

Example Code

In this example we query the position of the center of the pick aperture.

```
#define INCL_GPICORRELATION
#include <OS2.H>

BOOL flResult;
HPS hps; /* Presentation space handle. */
POINTL ptlRefPoint; /* Pick-aperture position. */
LONG xcoord, ycoord;
flResult = GpiPickAperturePosition(hps, &ptlRefPoint);
xcoord = ptlRefPoint.x; ycoord = ptlRefPoint.y;
```

GpiQueryPickApertureSize – Query Pick Aperture Size

```
#define INCL_GPICORRELATION /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryPickApertureSize (HPS hps, PSIZEL pszlSize)

This function returns the value of the pick-aperture size, as set by the GpiSetPickApertureSize function.

Parameters

hps (HPS) – input
Presentation-space handle.

pszlSize (PSIZEL) – output
Pick-aperture size.
Size of the pick aperture, in presentation-page coordinates.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Example Code

In this example we query the pick-aperture size, as set by the GpiSetPickApertureSize call.

```
#define INCL_GPICORRELATION
#include <OS2.H>

BOOL flResult;
HPS hps;          /* Presentation space handle. */
SIZEL szl;        /* Pick-aperture position. */
LONG xcoord, ycoord;
flResult = GpiQueryPickApertureSize(hps, &szl);
xcoord = szl.cx; ycoord = szl.cy;
```

GpiQueryPS – Query Presentation Space

```
#define INCL_GPICONTROL /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryPS (HPS hps, PSIZEL pszlSize)

This function returns page parameters for the presentation space.

Parameters

hps (HPS) – input
Presentation-space handle.

pszlSize (PSIZEL) – output
Presentation-page size.

Returns

Presentation-space options.

For details, see the GpiCreatePS function.

The individual fields of the presentation-space options can be extracted by ANDing the returned value with the appropriate constant.

The *PS_ASSOCIATE* field of *flOptions* (see GpiCreatePS) should not be used on this function. The value of this field is not necessarily the same value that is specified when the presentation space is created.

PS_UNITS	Presentation-space size units
PS_FORMAT	Presentation-space coordinate format
PS_TYPE	Presentation-space type
PS_MODE	Presentation-space mode.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.

Example Code

In this example we query the presentation space that corresponds to handle hps.

```
#define INCL_GPICONTROL
#include <OS2.H>
HPS hps;
SIZEL szl;
```

```
GpiQueryPS(hps, &szl);
```

GpiQueryRealColors – Query Real Colors

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryRealColors (HPS hps, ULONG uiOptions, LONG IStart, LONG ICount, PLONG aiColors)

This function returns the RGB values of the distinct colors available on the currently associated device.

Parameters

hps (HPS) – input

Presentation-space handle.

uiOptions (ULONG) – input

Options:

LCOLOPT_INDEX If this is specified, the index is to be returned for each RGB value.

If this flag is set when RGB mode is in force (LCOLF_RGB is set on GpiCreateLogColorTable), the RGB value is returned as the index.

Any color not available with the current logical color table is given a special index value of CLR_NOINDEX.

If it is not specified (flag is not set) index values are not returned.

Other

Other bits are reserved, and must be 0.

IStart (LONG) – input

Ordinal number of the first color required.

To start the sequence, this parameter is set to 0.

Note: This parameter is not the color index, and the order in which the colors are returned is not defined.

ICount (LONG) – input

Maximum number of elements.

Number of elements available in *aiColors*.

aiColors (PLONG) – output

Array in which the information is returned.

Contents depend on the setting of the LCOLOPT_INDEX flag:

0 An array of color values (each value is as defined for GpiCreateLogColorTable).

1 An array of alternating color indexes and values, in the order index1, value1, index2, value2,... indexn, valuen. An even number of elements is always returned in this case.

Returns

Number of elements returned:

≥0 Number of elements returned

GPI_ALTERERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

GpiQueryRealColors — Query Real Colors

PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_INV_COLOR_OPTIONS	An invalid options parameter was specified with a logical color table or color query function.
PMERR_INV_COLOR_START_INDEX	An invalid starting index parameter was specified with a logical color table or color query function.

Remarks

Subject to space in the *alColors* parameter, all colors that are physically available on the device are returned.

Use of the palette manager by other applications can effect the the physical colors available on the device. The available colors can change as a result of palette management, when this occurs a WM_REALIZEPALETTE message is sent to all applications.

Example Code

In this example we obtain the RGB values of the distinct colors available on the currently associated device.

```
#define INCL_GPILOGCOLORTABLE
#include <OS2.H>

LONG lResult;      /* number of elements returned */
HPS hps;           /* Presentation space handle. */
ULONG flOptions;    /* options */
LONG lStart;        /* ordinal number of first color */
LONG lCount;        /* maximum number of elements */
LONG alColors[5];   /* array containing return information */

flOptions = LCOLOPT_INDEX; /* return index for each RGB value. */
lStart = 0L;               /* start sequence at 0. */
lCount = 5L;               /* maximum of 5 elements. */

lResult = GpiQueryRealColors(hps,
                             flOptions,
                             lStart,
                             lCount,
                             alColors);
```

GpiQueryRegionBox – Query Region Box

```
#define INCL_GPIREGIONS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryRegionBox (HPS hps, HRGN hrgn, PRECTL prclBound)

This function returns the dimensions of the smallest rectangle able to bound the region.

Parameters

hps (HPS) – input

Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

hrgn (HRGN) – input

Region handle.

prclBound (PRECTL) – output

Bounding rectangle.

Returns

Complexity of region and error indicators:

RGN_NULL Null region

RGN_RECT Rectangular region

RGN_COMPLEX Complex region

RGN_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HRGN

An invalid region handle was specified.

PMERR_REGION_IS_CLIP_REGION

An attempt was made to perform a region operation on a region that is selected as a clip region.

PMERR_HRGN_BUSY

An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

Remarks

If the region is null, the rectangle returned has the left boundary equal to the right, and the top boundary equal to the bottom.

It is invalid if the specified region is currently selected as the clip region (by GpiSetClipRegion).

GpiQueryRegionBox — Query Region Box

Example Code

In this example we determine the dimensions of the smallest rectangle able to bound the region.

```
#define INCL_GPIPREGIONS
#include <OS2.H>

LONG lResult;      /* number of elements returned */
HPS hps;           /* Presentation space handle. */
HRGN hrgn;         /* region handle */
RECTL rc1Bound; /* bounding rectangle */

lResult = GpiQueryRegionBox(hps,
                           (VOID *)hrgn,
                           (PRECTL)&rc1Bound);
```

GpiQueryRegionRects – Query Region Rectangles

```
#define INCL_GPIREGIONS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryRegionRects (**HPS** hps, **HRGN** hrgn, **PRECTL** prclBound,
PRGNRECT prgnrcControl, **PRECTL** arclRects)

This function returns the rectangles that, when ORed together, define the specified region.

Parameters

hps (**HPS**) – input
Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

hrgn (**HRGN**) – input
Region handle.

prclBound (**PRECTL**) – input
Bounding rectangle.

NULL Return all the rectangles in the region.

Other Return only rectangles that intersect with the bounding rectangle. Each rectangle returned is the intersection of the bounding rectangle with a rectangle in the region.

prgnrcControl (**PRGNRECT**) – input/output
Processing-control structure.

arclRects (**PRECTL**) – output
Array of rectangle structures, in which the rectangles are returned.

The maximum number of rectangles that can be returned is specified by the *crc* parameter of the *RGNRECT* structure identified by the *prgnrcControl* parameter.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from *WinGetLastError*

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_HRGN	An invalid region handle was specified.
PMERR_INV_REGION_CONTROL	An invalid control parameter was specified with <i>GpiQueryRegionRects</i> .
PMERR_INV_COORDINATE	An invalid coordinate value was specified.
PMERR_INV_RECT	An invalid rectangle parameter was specified.
PMERR_REGION_IS_CLIP_REGION	An attempt was made to perform a region operation on a region that is selected as a clip region.
PMERR_HRGN_BUSY	An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

GpiQueryRegionRects — Query Region Rectangles

Remarks

Points on the right-hand and top boundaries are not included in the region. Points on the left-hand and bottom boundaries, that are not also on the right-hand or top boundaries (that is, the top-left and bottom-right corner points), are included.

It is invalid if the specified region is currently selected as the clip region (by GpiSetClipRegion).

Example Code

In this example we determine the rectangles that can be OR'ed together to determine the specified region.

```
#define INCL_GPIREGIONS
#include <OS2.H>
#define maxrects 12

BOOL flResult;      /* success indicator.      */
HPS hps;            /* presentation space handle. */
HRGN hrgn;          /* region handle.             */
RECTL rc1Bound;     /* bounding rectangle         */
RGNRECT rgnrcControl; /* processing control        */
RECTL arc1Rect[maxrects]; /* array of rectangle structures */
                        /* in which the rectangles are */
                        /* returned.                  */

rgnrcControl.ircStart = 1; /* start numbering rectangles */
                        /* from 1.                    */
rgnrcControl.crc = maxrects; /* maximum number of rectangles */
                        /* that can be returned.      */
rgnrcControl.usDirection = RECTDIR_LFRT_TOPBOT;
                        /* order rectangles left-to-right */
                        /* and top-to-bottom.          */
flResult = GpiQueryRegionRects(hps,
                                hrgn,
                                &rc1Bound, /* output */
                                &rgnrcControl,
                                /* prgnrcControl.crcReturned is the number */
                                /* of rectangles returned.          */
                                &arc1Rect[0]);
```

GpiQueryRGBColor – Query RGB Color

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryRGBColor (HPS hps, ULONG flOptions, LONG IColorIndex)

This function returns the actual RGB color that results from a particular index on the currently-associated device.

Parameters

hps (HPS) – input
Presentation-space handle.

flOptions (ULONG) – input
Options:

LCOLOPT_REALIZED

If this is specified, the information is required for when the logical color table is realized.

If it is not specified (flag is not set) the information is required for when the logical color table (if any) is not realized.

Other bits are reserved, and must be 0.

IColorIndex (LONG) – input
Color index.

This can be any normally valid color index value (see GpiSetColor) except CLR_DEFAULT.

Returns

RGB color providing closest match to the specified color index:

≥0 RGB color providing closest match

GPI_ALTEERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COLOR_OPTIONS

An invalid options parameter was specified with a logical color table or color query function.

PMERR_INV_COLOR_INDEX

An invalid color index parameter was specified with GpiQueryRGBColor.

Remarks

If an RGB logical color table has been loaded, this function returns the nearest RGB color. This function is then equivalent to GpiQueryNearestColor.

GpiQueryRGBColor — Query RGB Color

Example Code

This example uses the GpiQueryRGBColor call to determine if the color white is available.

```
#define INCL_GPILOGCOLORTABLE
#include <OS2.H>

LONG lResult;      /* closest match to the specified index */
HPS hps;           /* Presentation space handle. */
ULONG flOptions;   /* options */
LONG lColorIndex;  /* color index */
lColorIndex = CLR_WHITE;
flOptions = LCOLOPT_REALIZED;
               /* information is required for when the */
               /* logical color table is realized. */

lResult = GpiQueryRGBColor(hps,
                           flOptions,
                           lColorIndex );
```

GpiQuerySegmentAttrs – Query Segment Attributes

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQuerySegmentAttrs (HPS hps, LONG ISegId, LONG IAttribute)

This function returns the current value of the specified attribute as set by the GpiSetSegmentAttrs function.

Parameters

hps (HPS) – input
Presentation-space handle.

ISegId (LONG) – input
Segment identifier; must be greater than 0.

The name of the segment for which attribute information is to be returned.

IAttribute (LONG) – input
Attribute to be queried.

For details of the following attributes, see the GpiSetInitialSegmentAttrs function.

Identifies the attribute of the segment to be returned by this function:

ATTR_DETECTABLE	Detectability
ATTR_VISIBLE	Visibility
ATTR_CHAINED	Chained
ATTR_DYNAMIC	Dynamic
ATTR_FASTCHAIN	Fast chaining
ATTR_PROP_DETECTABLE	Propagate detectability
ATTR_PROP_VISIBLE	Propagate visibility.

Returns

Current attribute value:

ATTR_ON On/yes

ATTR_OFF Off/no

ATTR_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_SEG_NAME	An invalid segment identifier was specified.
PMERR_INV_SEG_ATTR	An invalid attribute parameter was specified with GpiSetSegmentAttrs, GpiQuerySegmentAttrs, GpiSetInitialSegmentAttrs, or GpiQueryInitialSegmentAttrs.
PMERR_SEG_NOT_FOUND	The specified segment identifier did not exist
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.

GpiQuerySegmentAttrs — Query Segment Attributes

Remarks

The segment can be any retained segment, including the currently open one if this is retained.

Example Code

This function is used to query the current value of the specified attribute.

```
#define INCL_GPISEGMENTS
#include <OS2.H>

LONG lSegid;    /* Segment identifier must */
LONG lValue;    /* be greater than 0. */
LONG lattribute; /* attribute to be queried */
HPS hps;        /* Presentation-space */
                /* handle. */

lattribute = ATTR_VISIBLE;

lValue = GpiQuerySegmentAttrs(hps,
                              lSegid,
                              lattribute);
```

GpiQuerySegmentNames – Query Segment Names

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQuerySegmentNames (HPS hps, LONG IFirstSegId, LONG ILastSegId, LONG IMax, PLONG alSegIds)

This function returns the identifiers of all segments that exist with identifiers in a specified range.

Parameters

hps (HPS) – input

Presentation-space handle.

IFirstSegId (LONG) – input

First segment in the range (must be greater than 0).

ILastSegId (LONG) – input

Last segment in the range (must be greater than 0).

IMax (LONG) – input

Maximum number.

This is the maximum number of segment identifiers to be returned in *alSegIds*.

alSegIds (PLONG) – output

Array in which the required identifiers are returned.

Returns

Number of identifiers returned:

≥0 Number of identifiers returned

GPI_ALTError Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME

An invalid segment identifier was specified.

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

PMERR_INV_MICROPS_FUNCTION

An attempt was made to issue a function that is invalid in a micro presentation space.

Remarks

Nonretained segment identifiers are not returned. If *IFirstSegId* is the same as, or greater than *ILastSegId*, the search terminates after querying only the segment with *IFirstSegId*.

GpiQuerySegmentNames — Query Segment Names

Example Code

This function returns the identifiers of all segments that exist within a specified range.

```
#define INCL_GPISEGMENTS
#include <OS2.H>
#define Maxsegs 5

LONG lRetCount;
HPS hps; /* Presentation-space */
        /* handle. */
LONG lFirstSegid; /* First segment in the */
                  /* range (must be greater */
                  /* than 0). */

LONG lLastSegid; /* Last segment in the */
                 /* range (must be greater */
                 /* than 0). */

LONG lMax; /* This is the maximum */
           /* number of segment */
           /* identifiers to be returned */
           /* in aSegids. */

LONG aSegids[Maxsegs]; /* Array in which the */
                       /* required identifiers are */
                       /* returned. */

lFirstSegid = 1;
lLastSegid = Maxsegs;
lMax = Maxsegs;

lRetCount = GpiQuerySegmentNames(hps,
                                  lFirstSegid,
                                  lLastSegid,
                                  lMax,
                                  aSegids);
```

GpiQuerySegmentPriority – Query Segment Priority

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQuerySegmentPriority (HPS hps, LONG IRefSegId, LONG IOrder)

This function returns the identifier of the named segment that is chained immediately before or after a specified reference segment.

Parameters

hps (HPS) – input

Presentation-space handle.

IRefSegId (LONG) – input

Reference-segment identifier.

IOrder (LONG) – input

Segment higher or lower.

Shows whether a segment identifier of a higher or lower priority than identified in the *IRefSegId* parameter is to be returned. Possible values are:

LOWER_PRI Return the next segment with a lower priority than *IRefSegId*. If *IRefSegId* = 0, query the identifier of the segment with the lowest priority.

HIGHER_PRI Return the next segment with a higher priority than *IRefSegId*. If *IRefSegId* = 0, query the identifier of the segment with the highest priority.

Returns

Segment identifier.

The identifier of the segment that is immediately before or after that specified in the *IRefSegId* parameter:

>0 Segment identifier.

0 The segment specified in the *IRefSegId* parameter is either the lowest-priority segment (when *IOrder* = **LOWER_PRI**) or the highest-priority segment (when *IOrder* = **HIGHER_PRI**).

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME

An invalid segment identifier was specified.

PMERR_INV_ORDERING_PARM

An invalid order parameter was specified with *GpiSetSegmentPriority*.

PMERR_SEG_NOT_CHAINED

An attempt was made to issue *GpiDrawFrom*, *GpiCorrelateFrom* or *GpiQuerySegmentPriority* for a segment that was not chained.

PMERR_SEG_NOT_FOUND

The specified segment identifier did not exist

PMERR_INV_MICROPS_FUNCTION

An attempt was made to issue a function that is invalid in a micro presentation space.

GpiQuerySegmentPriority — Query Segment Priority

Remarks

The segment that is chained before the specified segment, is considered to have a lower priority than the specified segment; similarly, the segment that is chained after the specified segment, is considered to have a higher priority than the specified segment.

Unnamed segments (with an identifier of zero) are ignored.

Example Code

This function returns the identifier of the named segment that is chained immediately before or after a specified reference segment.

```
#define INCL_GPISEGMENTS
#include <OS2.H>

HPS hps;          /* Presentation-space */
                  /* handle. */
LONG lSegid;      /* Segment identifier */
LONG lRefSegid;   /* Reference-segment */
                  /* identifier. */
LONG lOrder;      /* Shows whether a */
                  /* segment identifier of a */
                  /* higher or lower priority */
                  /* than identified in the */
                  /* lRefSegid parameter is */
                  /* to be returned. */

lOrder = HIGHER_PRI; /* Return the next */
                    /* segment with a higher */
                    /* priority than */
                    /* lRefSegid. If */
                    /* lRefSegid=0, query */
                    /* the identifier of the */
                    /* segment with the */
                    /* highest priority. */
lRefSegid = 0; /* find the segment with the highest */
              /* priority. */

lSegid = GpiQuerySegmentPriority(hps,
                                lRefSegid,
                                lOrder);
```

GpiQuerySegmentTransformMatrix – Query Segment Transform Matrix

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
```

```
BOOL GpiQuerySegmentTransformMatrix (HPS hps, LONG ISegId, LONG ICount,  
                                     PMATRIXLF pmatlfArray)
```

This function returns the elements of the transform of the identified segment (see GpiSetSegmentTransformMatrix).

Parameters

hps (HPS) – input
Presentation-space handle.

ISegId (LONG) – input
Segment identifier.

ICount (LONG) – input
Number of elements.

The number of elements that are to be set in the *pmatlfArray* parameter. *ICount* must be in the range 0 through 9.

pmatlfArray (PMATRIXLF) – output
Transform matrix.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME An invalid segment identifier was specified.

PMERR_INV_MICROPS_FUNCTION An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_LENGTH_OR_COUNT An invalid length or count parameter was specified.

PMERR_SEG_NOT_FOUND The specified segment identifier did not exist

GpiQuerySegmentTransformMatrix — Query Segment Transform Matrix

Example Code

This function returns the elements of the transform of the identified segment (see GpiSetSegmentTransformMatrix).

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
#include<OS2.H>
#define COUNT 9

HPS hps; /* Presentation-space */
/* handle. */
LONG lSegid; /* Segment identifier. */
LONG lCount; /* The number of elements */
/* that are to be set in the */
/* pmatlfArray parameter. */
/* lCount must be in the */
/* range 0 through 9. */
MATRIXLF pmatlfArray[COUNT]; /* array of Transform matrix */
/* structures. This is an */
/* output parameter. */
BOOL fSuccess; /* returns true if successful. */

fSuccess = GpiQuerySegmentTransformMatrix(hps,
                                           lSegid,
                                           lCount,
                                           pmatlfArray);
```

GpiQuerySetIds – Query Set Identifiers

```
#define INCL_GPILCIDS /* Or use INCL_GPI or INCL_PM */
```

```
BOOL GpiQuerySetIds (HPS hps, LONG ICount, PLONG alTypes, PSTR8 aNames,  
                    PLONG allcids)
```

This function returns information about all the fonts that have been created by GpiCreateLogFont, and tagged bit maps (see GpiSetBitmapId).

Parameters

hps (HPS) – input

Presentation-space handle.

ICount (LONG) – input

The number of objects to be queried.

The number of local identifiers (lcids) currently in use, and therefore the maximum number of objects for which information can be returned, can be found with GpiQueryNumberSetIds.

alTypes (PLONG) – output

Object types.

Elements indicate whether the corresponding *allcids* element refers to a logical font or a tagged bit map.

LCIDT_FONT Font object

LCIDT_BITMAP Bit map.

aNames (PSTR8) – output

Font names.

An array of *ICount* consecutive 8-byte fields, in which the 8-character names associated with the logical fonts are returned. For bit maps, the whole field is set to zeros.

allcids (PLONG) – output

Local identifiers.

An array in which the local identifier (lcid) values are returned.

LCID_DEFAULT is included if the default font has been changed (see GpiCreateLogFont).

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

GpiQuerySetIds — Query Set Identifiers

Remarks

Each of the output parameters is an array with *lCount* elements. Information about the first *lCount* objects is returned; if there are fewer than *lCount*, the *allTypes* and *allCids* elements for the remainder are cleared to 0.

Example Code

This example uses the `GpiQuerySetIds` function to retrieve the local identifier for all logical fonts. It then uses the identifiers to delete the logical fonts.

```
#define INCL_DOSMEMMGR
#define INCL_GPILCIDS
#include <OS2.H>
#define TOTALMEM 200

HPS hps;          /* Presentation-space */
                  /* handle. */
LONG lCount;       /* The number of objects to */
                  /* be queried. */
PLONG allTypes;    /* Object types. */
ULONG rc;          /* Return code. */
PSTR8 aNames;      /* font names. */
PLONG allCids;     /* local identifiers. */
PLONG pBase;
USHORT i;
rc = DosAllocMem((PPVOID)pBase,
                (ULONG)TOTALMEM*sizeof(LONG),
                /* space is needed for an array of */
                /* lCount longs. */
                PAG_READ |
                PAG_WRITE |
                PAG_COMMIT);

lCount = GpiQueryNumberSetIds(hps);
          /* The number of local */
          /* identifiers (lcids) */
          /* currently in use, and */
          /* therefore the maximum */
          /* number of objects for */
          /* which information can be */
          /* returned. */
rc = DosSubAllocMem((PVOID)pBase,
                   (PPVOID)aNames,
                   (ULONG)(lCount*(ULONG)sizeof(STR8)));
          /* space is needed for an array of */
          /* lCount longs. */
rc = DosSubAllocMem((PVOID)pBase,
                   (PPVOID)allCids,
                   (ULONG)lCount*sizeof(LONG));
          /* space is needed for an array of */
          /* lCount longs. */
rc = DosSubAllocMem((PVOID)pBase,
                   (PPVOID)allTypes,
                   (ULONG)lCount*sizeof(LONG));
          /* space is needed for an array of */
          /* lCount longs. */
GpiQuerySetIds(hps,
               lCount,
               allTypes,
               aNames, /* An array of lCount */
                      /* consecutive 8-byte fields, */
                      /* in which the 8-character */
                      /*
```

GpiQuerySetIds — Query Set Identifiers

```
/* names associated with */
/* the logical fonts are */
/* returned. For bit maps, */
/* the whole field is set to */
/* zeros. */
allcids);/* An array in which the */
/* local identifier (lcid) */
/* values are returned. */
/* LCID_DEFAULT is */
/* included if the default */
/* font has been changed */
/* (see GpiCreateLogFont). */

for (i = 1; i < lCount; i++)
{
    if (allcids[i] == LCIDT_FONT)
        GpiDeleteSetId(hps,allcids[i]);
}
```

GpiQueryStopDraw — Query Stop Draw

```
#define INCL_GPICONTROL /* Or use INCL_GPI or INCL_PM */
```

LONG GpiQueryStopDraw (HPS hps)

This function indicates whether the “stop draw” condition currently exists.

Parameters

hps (HPS) — input
Presentation-space handle.

Returns

Stop draw condition indicator:

SDW_OFF No “stop draw” condition currently exists
SDW_ON The “stop draw” condition currently exists
SDW_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.

Remarks

See GpiSetStopDraw for details.

Example Code

This function indicates whether the “stop draw” condition currently exists.

```
#define INCL_GPICONTROL
#include <OS2.H>
HPS hps;          /* Presentation-space */
                  /* handle.          */
LONG lValue;

if(GpiQueryStopDraw(hps) == SDW_OFF)
{
    /* drawing may proceed; no stop draw */
    /* condition exists.                  */
}
```

```
#define INCL_GPICORRELATION /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryTag (HPS hps, PLONG piTag)

This function returns the current value of the tag identifier, as set by the GpiSetTag function.

Parameters

hps (HPS) – input.
Presentation-space handle.

piTag (PLONG) – output
Tag identifier.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION

An attempt was made to issue a function that is invalid in a micro presentation space.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Example Code

This function returns the current value of the tag identifier, as set by the GpiSetTag call.

```
#define INCL_GPICORRELATION
#include <OS2.H>

HPS hps;          /* Presentation-space */
                  /* handle.          */

LONG iTag;        /* Tag identifier. */

GpiQueryTag(hps,
            &iTag);
```


GpiQueryTextAlignment — Query Text Alignment

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryTextAlignment (HPS hps, PLONG piHorizontal, PLONG piVertical)

This function returns the current value of the text alignment attribute, as set by the GpiSetTextAlignment function.

Parameters

hps (HPS) — input

Presentation-space handle.

piHorizontal (PLONG) — output

Horizontal alignment: The horizontal alignment determines character positioning in a text string. The value returned will be one of those described under the GpiSetTextAlignment function.

piVertical (PLONG) — output

Vertical alignment: The vertical alignment determines character positioning in a text string. The value returned will be one of those described under the GpiSetTextAlignment function.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE

An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE

An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Support for this function is device dependent.

Related Functions

- GpiQueryAttrs
- GpiSetTextAlignment

GpiQueryTextBox – Query Text Box

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

```
BOOL GpiQueryTextBox (HPS hps, LONG ICount1, PCH pchString, LONG ICount2,  
PPOINTL aptlPoints)
```

This function returns the relative coordinates of the four corners of a text box.

Parameters

hps (HPS) – input
Presentation-space handle.

ICount1 (LONG) – input
Number of characters.

pchString (PCH) – input
The character string.

ICount2 (LONG) – input
Number of points.

Contains the number of points to be returned in the *aptlPoints* array. Specify `TEXTBOX_COUNT` to get the maximum information.

aptlPoints (PPOINTL) – output
List of points.

The list of points contains the relative coordinates of the text box in world coordinates. The array elements are numbered consecutively, starting with `TEXTBOX_TOPLEFT`. The element number constants start with 0. Refer to the appropriate bindings reference. A *ICount2* value of `TEXTBOX_COUNT` will cause all of the defined array elements to be returned.

The terms 'top-left', 'bottom-right', and so on, are well defined when the character angle is such that the baseline is parallel to the x axis and running left to right, and there is no character shear. If the character string is rotated or sheared, the term top-left applies to the corner of the box that appears in the top-left position when no rotation or shear is applied.

This is an example:

Set character angle = -1,1

String = ABCDE

Coordinates returned are as shown:

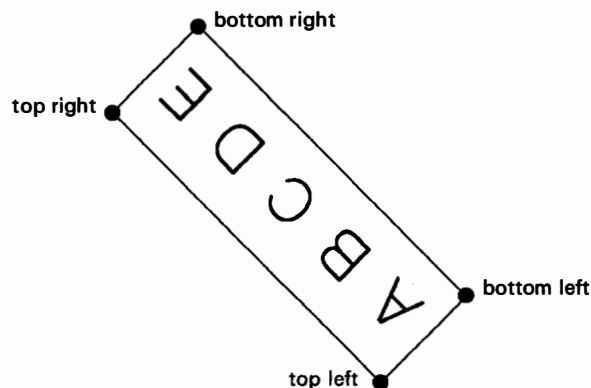


Figure 5-4. Box Enclosing Characters

GpiQueryTextBox —

Query Text Box

TXTBOX_TOPLEFT	Top-left corner
TXTBOX_BOTTOMLEFT	Bottom-left corner
TXTBOX_TOPRIGHT	Top-right corner
TXTBOX_BOTTOMRIGHT	Bottom-right corner
TXTBOX_CONCAT	Concatenation point.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_COORDINATE_OVERFLOW	An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

The text box is defined as the parallelogram that encloses the specified character string when displayed on the device. Also returned are the relative coordinates of the concatenation point; that is, the value of current position after an equivalent GpiCharStringAt function. All coordinates are relative to the start point. (See GpiSetCharDirection function.) These coordinates can be used to box or underline the string, or to change the attributes in the middle of a longer string.

Note: The height of the string is based on the maximum height of the font (including space for descenders, accents, and so on), not the maximum height of the actual characters in the string. The dimensions of the box do not correspond directly to those of the character box (see GpiSetCharBox).

Character attributes are taken into account as if the string is to be drawn, but no output actually occurs. However, if the character mode (see GpiSetCharMode) is CM_MODE2 this function should only be used if the character angle (see GpiSetCharAngle), character direction (see GpiSetCharDirection) and character shear (see GpiSetCharShear) attributes are set to their default values.

This function is not valid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Example Code

This example uses the GpiQueryTextBox function to draw a line under the string. The GpiCharString function draws the string at the point (100,100). Since the points retrieved by GpiQueryTextBox are relative to the start of the string, the starting point needs to be added to the points that are used to draw the underline.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;
POINTL apt1[TXTBOX_COUNT];
POINTL pt1 = { 100, 100 };

GpiQueryTextBox(hps,
                11L,
                "This string",
                TXTBOX_COUNT, /* return maximum information */
                apt1);        /* array of coordinates points */
                               /* in world coordinates. */

apt1[1].x += pt1.x;
apt1[1].y += pt1.y;
GpiMove(hps, &apt1[1]);
apt1[3].x += pt1.x;
apt1[3].y += pt1.y;
GpiLine(hps, &apt1[3]);
GpiMove(hps, &pt1);
GpiCharString(hps, 11L, "This string");
```

GpiQueryViewingLimits –

Query Viewing Limits

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryViewingLimits (HPS hps, PRECTL prclLimits)

This function returns the current value of the viewing limits, as set by the GpiSetViewingLimits function.

Parameters

hps (HPS) – input
Presentation-space handle.

prclLimits (PRECTL) – output
Viewing limits.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not draw or draw-and-retain .
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryViewingLimits – Query Viewing Limits

Example Code

In this example the model space clipping region width is reduced to 100 if it is greater.

```
#define INCL_GPITRANSFORMS
#include <OS2.H>

HPS hps;          /* Presentation-space */
                  /* handle.          */
RECTL rcLimits; /* viewing limits.    */
BOOL fSuccess;

fSuccess = GpiQueryViewingLimits(hps,
                                &rcLimits);

if ((rcLimits.xRight - rcLimits.xLeft) > 100)
{
    rcLimits.xRight = 100;
    rcLimits.xLeft = 200;
}

fSuccess = GpiSetViewingLimits(hps,
                               &rcLimits);
```

GpiQueryViewingTransformMatrix —

Query Viewing Transform Matrix

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryViewingTransformMatrix (HPS hps, LONG ICount, PMATRIXLF pmatlfArray)

This function returns the current viewing transform (see GpiSetViewingTransformMatrix)..

Parameters

hps (HPS) — input

Presentation-space handle.

ICount (LONG) — input

Number of elements.

The number of elements to be returned in *pmatlfArray* (must be in the range 0 through 9). If 0 is specified, no matrix elements are returned.

pmatlfArray (PMATRIXLF) — output

Transform matrix.

A structure in which the elements of the viewing transform matrix are returned.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION

An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

GpiQueryViewingTransformMatrix – Query Viewing Transform Matrix

Example Code

This example uses the GpiQueryViewingTransformMatrix function to see if the width and the height of drawing are already doubled. If this is not the case, the GpiSetViewingTransformMatrix is used to replace the existing viewing transformation. The new transformation will then double the width and height of drawing.

```
#define INCL_GPITRANSFORMS
#include <OS2.H>

HPS hps;          /* Presentation space handle. */
LONG lCount;      /* maximum number of elements */
MATRIXLF matlf = { MAKEFIXED(2,0), /* scale x coordinates by a */
                  /* factor of 2. */
                  0, 0, 0, /* no rotation. */
                  MAKEFIXED(2,0), /* scale y coordinates by a */
                  /* factor of 2. */
                  0, 0, 0, 1}; /* no rotation. */
lCount = 9L;      /* number of elements. */
GpiQueryViewingTransformMatrix(hps,
                              lCount,
                              &matlf);

if (matlf.fxM12 == MAKEFIXED(2, 0))
{
    GpiSetViewingTransformMatrix(hps,
                                lCount,
                                &matlf,
                                TRANSFORM_REPLACE);
}
```


GpiQueryWidthTable — Query Font Width Table

```
#define INCL_GPILCIDS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiQueryWidthTable (HPS hps, LONG IFirstChar, LONG ICount, PLONG aiData)

This function returns width table information for the logical font identified by the value of the character-set attribute.

Parameters

hps (HPS) — input

Presentation-space handle.

IFirstChar (LONG) — input

Codepoint of first character.

The codepoint of the initial character, for which width-table information is required.

ICount (LONG) — input

Count of elements in *aiData*.

The number that should be allowed for, so as to retrieve the full width table. Data for this font can be found by GpiQueryFontMetrics.

aiData (PLONG) — output

Array of width values.

An array of *ICount* elements, in which width-table information is returned. No more than *ICount* elements are returned.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_FIRST_CHAR

An invalid firstchar parameter was specified with GpiQueryWidthTable.

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

PMERR_COORDINATE_OVERFLOW

An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

GpiQueryWidthTable — Query Font Width Table

Example Code

In this example the widths of the first 50 characters of the current font are obtained.

```
#define INCL_GPILCIDS
#include <OS2.H>
#define COUNT 50
```

```
HPS hps;          /* Presentation-space */
                  /* handle. */
LONG alData[COUNT]; /* array of width values. */
```

```
GpiQueryWidthTable(hps,
                   0,
                   COUNT,
                   alData);
```

GpiRectInRegion – Rectangle In Region

```
#define INCL_GPIREGIONS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiRectInRegion (HPS hps, HRGN hrgn, PRECTL prclRect)

This function checks whether any part of a rectangle lies within the specified region.

Parameters

hps (HPS) – input

Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

hrgn (HRGN) – input

Region handle.

prclRect (PRECTL) – input.

Test rectangle.

The rectangle is specified in device coordinates.

Returns

Inside and error indicators:

RRGN_OUTSIDE Not in region

RRGN_PARTIAL Some in region

RRGN_INSIDE All in region

RRGN_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HRGN

An invalid region handle was specified.

PMERR_INV_COORDINATE

An invalid coordinate value was specified.

PMERR_INV_RECT

An invalid rectangle parameter was specified.

PMERR_REGION_IS_CLIP_REGION

An attempt was made to perform a region operation on a region that is selected as a clip region.

PMERR_HRGN_BUSY

An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

Remarks

It is invalid if the specified region is currently selected as the clip region (by GpiSetClipRegion).

GpiRectInRegion – Rectangle In Region

Related Functions

- GpiCombineRegion
- GpiCreateRegion
- GpiDestroyRegion
- GpiEqualRegion
- GpiOffsetRegion
- GpiPaintRegion
- GpiPtInRegion
- GpiQueryRegionBox
- GpiQueryRegionRects
- GpiSetRegion

Example Code

In this example we check to see if a rectangle is inside a region before we destroy the region.

```
#define INCL_GPIREGIONS
#include <OS2.H>
HPS hps;          /* presentation-space handle. */
HRGN hrgn;        /* region handle. */
PRECTL prclRect; /* test rectangle. */
LONG lInside;     /* result. */
lInside = GpiRectInRegion(hps,
                          hrgn,
                          prclRect);

if (lInside == RRGN_OUTSIDE)
{
    GpiDestroyRegion(hps, hrgn);
}
```

GpiRectVisible – Rectangle Visible

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

LONG GpiRectVisible (HPS hps, PRECTL prclRectangle)

This function checks whether any part of a rectangle lies within the clipping region of the device associated with the specified presentation space.

Parameters

hps (HPS) – input

Presentation-space handle.

prclRectangle (PRECTL) – input

Test rectangle, in world coordinates.

Points on the borders of the rectangle are considered to be included within the rectangle.

Returns

Visibility indicator:

RVIS_INVISIBLE Not visible

RVIS_PARTIAL Some of the rectangle is visible

RVIS_VISIBLE All of the rectangle is visible

RVIS_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE An invalid coordinate value was specified.

PMERR_INV_RECT An invalid rectangle parameter was specified.

Remarks

For the purposes of this function, the clipping region is defined as the intersection between the application clipping region and any other clipping, including windowing.

Related Functions

- GpiExcludeClipRectangle
- GpiIntersectClipRectangle
- GpiOffsetClipRegion
- GpiPtVisible
- GpiQueryClipBox
- GpiQueryClipRegion
- GpiSetClipRegion
- WinExcludeUpdateRegion

GpiRectVisible – Rectangle Visible

Example Code

In this example the GpiRectVisible call is used to determine if all of the rectangle is visible.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;           /* presentation-space handle. */
LONG lVisibility;   /* visibility indicator */
PRECTL prclRectangle; /* test rectangle in world */
                  /* coordinates. */

lVisibility = GpiRectVisible(hps,
                             prclRectangle);

if (lVisibility == RVIS_INVISIBLE) /* rectangle is not */
{                                  /* visible. */
    /* code block */
}
```

GpiRemoveDynamics — Remove Dynamics

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiRemoveDynamics (HPS hps, LONG IFirstSegId, LONG ILastSegId)

This function removes those parts of the displayed image that are drawn from the dynamic segments in a section of the segment chain. This includes any parts that are drawn by calls from these dynamic segments.

Parameters

hps (HPS) — input
Presentation-space handle.

IFirstSegId (LONG) — input
First segment in the section.
It must be greater than 0.

ILastSegId (LONG) — input
Last segment in the section.
It must be greater than 0.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.
PMERR_INV_SEG_NAME	An invalid segment identifier was specified.
PMERR_INV_FOR_THIS_DC_TYPE	An attempt has been made to issue GpiRemoveDynamics or GpiDrawDynamics to a presentation space associated with a metafile device context.

Remarks

This function usually indicates that a dynamic segment is about to be updated; and that, having completed the update, GpiDrawDynamics is called to redraw the dynamic segments.

If there is more than one dynamic segment, only those that are being updated need be removed. The section of the segment chain is identified by the first and last segments in the section. If *IFirstSegId* and *ILastSegId* have the same value, this call erases only the parts drawn from the segment, and by calls from that segment.

Specifying the range of segment identifiers that are to be removed usually has a performance advantage, in that searching of the chain stops after *ILastSegId* has been processed. It can also be used to operate on less than the maximum number of dynamic segments, as in one of the following examples:

GpiRemoveDynamics – Remove Dynamics

- Several dynamic segments are currently drawn, but only one is to be updated. Identifying this segment with both *IFirstSegid* and *ILastSegid* means that only this one is removed. It can then be updated, and replaced with *GpiDrawDynamics*.
- A new dynamic segment can be created, while the rest remain drawn. *GpiRemoveDynamics* is issued before the segment has been created (or while it is unchained, if it already exists), identifying it with both *IFirstSegid* and *ILastSegid*. It is then created with this identifier (or chained, if it already exists), and *GpiDrawDynamics* is issued, causing it to be drawn.

In these examples, the other dynamic segments remain drawn throughout.

In all cases, after *GpiDrawChain*, *GpiDrawDynamics*, *GpiDrawFrom*, or *GpiDrawSegment*, where the *DCTL_DYNAMIC* draw control is set (see *GpiSetDrawControl*), *all* dynamic segments must be drawn. The *IFirstSegid* and *ILastSegid* parameters of *GpiRemoveDynamics*, cannot be used to cause a subset of dynamic segments to be drawn after the following *GpiDrawDynamics*. If this is required, it can be done by unchaining the unwanted dynamic segments after first removing them.

Dynamic segments that are currently drawn must never be updated in the segment store, nor must any drawing in mix modes (other than exclusive-OR or leave-alone) be done to a presentation space while dynamic segments are drawn in it.

If a temporary re-association is to be done, this function must be issued to remove the dynamic segments from the display before the first dissociation.

It is necessary to ensure that attributes, model transform, current position, and viewing limits are reset to their default values, before processing the segments. This can either be done by ensuring that the first dynamic segment in the removed section does not have the *ATTR_FASTCHAIN* attribute (see *GpiSetInitialSegmentAttrs*), or by issuing *GpiResetPS* before the *GpiRemoveDynamics*. The latter method also resets the clip path to cause no clipping, which may also be necessary.

If this function is followed by primitives or attributes, without first opening a segment, the processing is as described for *GpiCloseSegment*. In particular, note that during *GpiRemoveDynamics*, the system forces the foreground mix to *FM_XOR* and the background mix to *BM_LEAVEALONE*. It may be necessary to set one or both of these before starting to draw.

If *IFirstSegid* does not exist, or is not in the segment chain, no removal or drawing occurs. However, the segment identifier range is still established for a subsequent *GpiDrawDynamics* function.

If *ILastSegid* does not exist, or is not in the chain, or is chained before *IFirstSegid*, no error is raised, and processing continues to the end of the chain.

Related Functions

- *GpiDrawChain*
- *GpiDrawDynamics*
- *GpiDrawFrom*
- *GpiDrawSegment*
- *GpiErase*
- *GpiGetData*
- *GpiPutData*
- *GpiSetDrawControl*
- *GpiSetDrawingMode*
- *GpiSetStopDraw*

GpiRemoveDynamics — Remove Dynamics

Example Code

This example uses the GpiRemoveDynamics function to remove the image drawn by the dynamic segment whose segment identifier is 4. It then edits the segment and redraws it, using the GpiDrawDynamics function.

```
#define INCL_GPISEGMENTS
#define INCL_GPICONTROL
#include <OS2.H>

POINTL pt1 = {30, 40};
HPS hps; /* presentation space handle */

/* Remove the image for dynamic segment #4. */

GpiRemoveDynamics(hps, 4L, 4L);

/* Edit the segment. */

GpiSetDrawingMode(hps, DM_RETAIN);
GpiOpenSegment(hps, 4L);
GpiSetElementPointer(hps, 1L);
GpiMove(hps, &pt1);
GpiCloseSegment(hps);

GpiDrawDynamics(hps); /* redraws the edited segment */
```

GpiResetBoundaryData – Reset Boundary Data

```
#define INCL_GPICORRELATION /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiResetBoundaryData (HPS hps)

This function resets the boundary data to null.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

Remarks

This function is only necessary for **draw** mode (see GpiSetDrawingMode) boundary determination. Boundary data is automatically reset before any retained drawing call.

After drawing, boundary data can be found by issuing GpiQueryBoundaryData.

Note: Boundary data is not reset at the start of a segment.

Related Functions

- GpiQueryBoundaryData
- GpiSetDrawControl

Example Code

This function is used to reset the boundary data to null. It is only necessary for draw mode boundary determination.

```
#define INCL_GPICORRELATION
#include <OS2.H>
```

```
HPS hps;          /* presentation space handle */
```

```
GpiResetBoundaryData(hps);
```

GpiResetPS —

Reset Presentation Space

```
#define INCL_GPICONTROL /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiResetPS (HPS hps, ULONG flOptions)

This function resets the presentation space.

Parameters

hps (HPS) — input
Presentation-space handle.

flOptions (ULONG) — input
Reset option:

GRES_ATTRS

This has the following effects:

- All current attributes and arc parameters are reset to their default values
- The current tag is reset to its default value
- The current model transform is reset to unity
- The current position is set to (0,0)
- Any open path or area is aborted
- Any open element bracket is closed
- Any open segment is closed
- The current clip path is set so as to cause no clipping
- The current viewing limits are reset to their default values.

GRES_SEGMENTS

This has all the effects of GRES_ATTRS plus:

- Any retained segments are deleted
- Initial segment attributes are reset to their default values
- The default viewing transform and the graphics field are reset to their default values
- The viewing transform is set to unity
- Drawing mode, draw controls, edit mode and attribute mode are reset to default values
- Boundary data is reset
- The currently selected clip region, if any, is deselected, and destroyed
- The default values of primitive attributes, arc parameters, viewing limits and primitive tag are reset to their initial values.

GRES_ALL

This has all the effects of GRES_ATTRS and GRES_SEGMENTS plus:

- Any logical fonts and local identifiers for bit maps are deleted (the default character set is restored if it has been changed).
- Any loaded logical color table is reset to default.
- Any palette selected into the presentation space (see GpiSelectPalette) is deselected.
- The pick aperture size and position are reset to default.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

GpiResetPS – Reset Presentation Space

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_RESET_OPTIONS

An invalid options parameter was specified with GpiResetPS.

Remarks

Three levels of reset are provided. These are, in increasing order of power:

- As if a new (root) segment is being processed
- As if the presentation space is being created without deleting resources
- As if the presentation space is being created with resources deleted.

More details are provided under the description of *fIOptions* above.

None of these options cause any drawing or erasing to take place on the device (GpiErase can be used to do the latter), nor is any association between the specified presentation space and a device context affected. The page viewport is also unaffected.

After restoring a presentation space that has a palette selected into it, WinRealizePalette must be issued before any drawing calls or calls to query colors.

Note: This function must not be used when creating SAA-conforming metafiles; see “Metafile Restrictions” on page G-1.

Related Functions

- GpiAssociate
- GpiCreatePS
- GpiDestroyPS
- GpiQueryDevice
- GpiQueryPS
- GpiRestorePS
- GpiSavePS
- GpiSetPS

Example Code

This function is used to reset the presentation space.

```
#define INCL_GPICONTROL
#include <OS2.H>

HPS hps;          /* presentation space handle */
ULONG fIOptions; /* reset options */

fIOptions = GRES_ALL; /* reset all options. */

GpiResetPS(hps, fIOptions);
```

GpiRestorePS — Restore Presentation Space

```
#define INCL_GPICONTROL /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

BOOL GpiRestorePS (HPS hps, LONG IPSid)

This function restores the state of the presentation space to the one that exists when the corresponding GpiSavePS is issued.

Parameters

hps (HPS) — input

Presentation-space handle.

IPSid (LONG) — input

Identifier of the saved presentation space that is to be restored:

If an error is returned, the stack is unchanged, as is the current presentation space.

>0 *IPSid* must be the identifier of a saved presentation space on the stack. It is an error if it does not exist.

0 Is an error. (This might have resulted from an invalid use of GpiSavePS).

<0 The absolute value of *IPSid* indicates how many saved presentation spaces on the stack are required. Thus **−1** means that the most recently saved one is to be restored. It is an error if the absolute value is larger than the number of entries on the stack.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_NOT_IN_DRAW_MODE

An attempt was made to issue GpiSavePS or GpiRestorePS while the drawing mode was not set to DM_DRAW.

PMERR_INV_ID

An invalid *IPSid* parameter was specified with GpiRestorePS.

Remarks

The most recently saved presentation space need not be the one that is restored. In this case, any that are skipped over on the stack are discarded.

Any clip regions selected into discarded presentation spaces are automatically destroyed.

This function is valid in an open element bracket and in an open segment bracket if the drawing mode (see GpiSetDrawingMode) is set to **draw** and within an open element bracket. If it occurs within an open area or path bracket, the corresponding GpiSavePS must have taken place earlier in the same bracket.

GpiRestorePS – Restore Presentation Space

Related Functions

- GpiAssociate
- GpiCreatePS
- GpiDestroyPS
- GpiQueryDevice
- GpiQueryPS
- GpiResetPS
- GpiSavePS
- GpiSetPS
- GpiPop

Example Code

This example restores the state of the presentation space to the one that exists when the corresponding GpiSavePS is issued.

```
#define INCL_GPICONTROL
#include <OS2.H>
HPS hps;      /* presentation space handle */
LONG lPSid;   /* the identifier of a saved presentation */
              /* space on the stack. */
GpiRestorePS(hps, lPSid);
```

GpiRotate — Rotate Transform

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiRotate (HPS hps, PMATRIXLF pmatlfArray, LONG IOptions, FIXED fxAngle, PPOINTL pptlCenter)

This function applies a rotation to a transform matrix.

Parameters

hps (HPS) — input
Presentation-space handle.

pmatlfArray (PMATRIXLF) — input/output
Transform matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type **FIXED**, and have an assumed binary point between the second and third bytes. Thus a value of 1.0 is represented by 65 536. Other elements are normal signed integers.

The third, sixth, and ninth elements must be 0, 0, and 1, respectively.

IOptions (LONG) — input
Transform options.

Specifies how the transform defined by the specified rotation should be used to modify the previous transform specified by the *pmatlfArray* parameter. Possible values are:

TRANSFORM_REPLACE The previous transform is discarded and replaced by the transform describing the specified rotation.

TRANSFORM_ADD The previous transform is combined with a transform representing the specified rotation in the order (1) previous transform, (2) rotational transform. This option is most useful for incremental updates to transforms.

fxAngle (FIXED) — input
Rotation angle.

The angle describing the rotation, measured counterclockwise from the x-axis in degrees.

pptlCenter (PPOINTL) — input
Center of rotation.

The point about which the rotation occurs.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_TRANSFORM_TYPE An invalid options parameter was specified with a transform matrix function.

GpiRotate – Rotate Transform

Remarks

This function is a helper function that either applies a specified rotational component to an existing transform matrix, or replaces the matrix with one that represents the specified rotation alone.

The transform is specified as a one-dimensional array of 9 elements that are the elements of a 3-row by 3-column matrix ordered by rows. The order of the elements is as follows:

Matrix	Array
$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix}$	$(a, b, 0, c, d, 0, e, f, 1)$

Transforms act on the coordinates of primitives, so that a point with coordinates (x,y) is transformed to the point:

$$(a*x + c*y + e, b*x + d*y + f)$$

The transform can be used in any call following:

- GpiSetModelTransformMatrix
- GpiSetSegmentTransformMatrix
- GpiSetViewingTransformMatrix
- GpiSetDefaultViewMatrix.

Other similar helper functions are:

- GpiTranslate to apply a translation component
- GpiScale to apply a scaling component.

Related Functions

- GpiScale
- GpiTranslate
- GpiSetModelTransformMatrix
- GpiSetSegmentTransformMatrix
- GpiSetDefaultViewMatrix
- GpiSetViewingTransformMatrix

GpiRotate — Rotate Transform

Example Code

In this example, the viewing transform matrix is rotated 10 degrees counterclockwise from the x-axis. Hence, everything will appear rotated.

```
#define INCL_GPITRANSFORMS
#include <OS2.H>
```

```
HPS hps;           /* presentation space handle */
MATRIXLF matlf;    /* transform matrix.          */
POINTL ptlCenter;  /* center of rotation.        */
```

```
GpiQueryViewingTransformMatrix(hps,
                                1L,
                                &matlf);

ptlCenter.x = 50L;
ptlCenter.y = 50L;
```

```
GpiRotate(hps,
           &matlf,
           TRANSFORM_REPLACE,
           MAKEFIXED(10,0), /* rotate 10 degrees left. the angle */
                               /* must be passed in fixed format. */
                               /* see the pmgpi.h file for a */
                               /* description of the MAKEFIXED macro. */
           &ptlCenter);
```

GpiSaveMetaFile – Save Metafile

```
#define INCL_GPIMETAFILES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSaveMetaFile (HMF hmf, PSZ pszFilename)

This function saves a metafile in a disk file.

Parameters

hmf (HMF) – input
Metafile handle.

pszFilename (PSZ) – input
File name.

The name of the file to which the metafile is to be saved. This name must be a valid external name.

It is an error if a file of this name exists already.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HMF

An invalid metafile handle was specified.

PMERR_DOSOPEN_FAILURE

A DosOpen call made during GpiLoadMetaFile or GpiSaveMetaFile gave a good return code but the file was not opened successfully.

PMERR_INSUFFICIENT_DISK_SPACE

The operation terminated through insufficient disk space.

PMERR_METAFILE_IN_USE

An attempt has been made to access a metafile that is in use by another thread.

PMERR_TOO_MANY_METAFILES_IN_USE

The maximum number of metafiles allowed for a given process was exceeded.

Remarks

The metafile is deleted from storage; this means that the metafile handle is no longer valid.

The metafile may be reaccessed by GpiLoadMetaFile.

Related Functions

- GpiCopyMetaFile
- GpiDeleteMetaFile
- GpiLoadMetaFile
- GpiPlayMetaFile
- GpiQueryMetaFileBits
- GpiQueryMetaFileLength
- GpiSetMetaFileBits

GpiSaveMetaFile — Save Metafile

Example Code

This function saves a metafile in a disk file.

```
#define INCL_GPIMETAFILES
#include <OS2.H>

HMF hmf;    /* metafile handle. */
GpiSaveMetaFile(hmf, "file.met");
```

GpiSavePS – Save Presentation Space

```
#define INCL_GPICONTROL /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

LONG GpiSavePS (HPS hps)

This function saves information about the presentation space on a LIFO (last in, first out) stack.

Parameters

hps (HPS) – input
Presentation-space handle.

Returns

Identifier of saved presentation space.

This may be used on a subsequent GpiRestorePS call. The identifier is equal to the depth of the saved presentation space on the save/restore stack, with 1 representing the base level:

≥1 Identifier of saved presentation space

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_NOT_IN_DRAW_MODE

An attempt was made to issue GpiSavePS or GpiRestorePS while the drawing mode was not set to DM_DRAW.

Remarks

The stack is different from the one used to save attribute values (see GpiSetAttrMode) in a normal presentation space.

This function, and GpiRestorePS, can be used with a micro presentation space, as well as a normal presentation space (in **draw** drawing mode only).

The presentation space itself is unchanged.

The following are saved:

- Current attributes
- Current transforms, viewing limits, and clip path
- Current position
- Reference to selected clip region
- Any loaded logical color table
- References to any loaded logical fonts
- References to the regions created on the associated device context.

The following are not saved:

- Draw controls
- Drawing mode
- Edit mode and attribute mode
- The visible region.

GpiSavePS —

Save Presentation Space

Note: Only references to resources, rather than the actual resources (such as clip region, logical fonts, and regions) are copied by this function, so the actual resources must not be changed.

This function is valid in an open segment bracket, but only if the drawing mode (see GpiSetDrawingMode) is set to **draw**. This function can occur within an open element bracket. When it occurs within an open area or path bracket, GpiRestorePS must be called before the bracket is closed.

If this function occurs during the generation of a metafile, the drawing mode must be set to **draw** when the metafile is replayed.

Related Functions

- GpiAssociate
- GpiCreatePS
- GpiDestroyPS
- GpiQueryDevice
- GpiQueryPS
- GpiResetPS
- GpiRestorePS
- GpiSetPS

Example Code

This example uses the GpiSavePS function to save the state of the presentation space. The identifier returned by the function is used in the call to the GpiRestorePS function to restore the saved state.

```
#define INCL_GPICONTROL
#include <OS2.H>

HPS hps; /* presentation-space handle. */
LONG idPS;

idPS = GpiSavePS(hps); /* saves the presentation-space state */
.
.
.
/* restores the presentation-space state */
GpiRestorePS(hps, idPS);
```

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiScale (HPS hps, PMATRIXLF pmatlfArray, LONG IOptions, PFIXED afxScale, PPOINTL pptlCenter)

This function applies a scaling component to a transform matrix.

Parameters

hps (HPS) – input
Presentation-space handle.

pmatlfArray (PMATRIXLF) – input/output
Transform matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type FIXED, and have an assumed binary point between the second and third bytes. Thus a value of 1.0 is represented by 65 536. Other elements are normal signed integers.

The third, sixth, and ninth elements must be 0, 0, and 1, respectively.

IOptions (LONG) – input
Transform options.

Specifies how the transform defined by the specified scaling should be used to modify the previous transform specified by the *pmatlfArray* parameter. Possible values are:

TRANSFORM_REPLACE The previous transform is discarded and replaced by the transform describing the specified scaling.

TRANSFORM_ADD The previous transform is combined with a transform representing the specified scaling in the order (1) previous transform, (2) scaling transform. This option is most useful for incremental updates to transforms.

afxScale (PFIXED) – input
Scale factors.

The first element of the array is the x scale factor, and the second is the y scale factor.

Scaling values outside the range -1 through +1 are not valid for subsequent use with presentation spaces that have a coordinate format (as set by the GpiCreatePS function) of GPIF_SHORT.

pptlCenter (PPOINTL) – input
Center of scale.

The point about which the scale occurs.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_TRANSFORM_TYPE

An invalid options parameter was specified with a transform matrix function.

GpiScale — Scale Matrix

Remarks

This function is a helper function which either applies a specified scaling component to an existing transform matrix, or replaces the matrix with one that represents the specified scaling alone.

The transform is specified as a one-dimensional array of 9 elements that are the elements of a 3-row by 3-column matrix ordered by rows. The order of the elements is:

Matrix

Array

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix}$$

(a,b,0,c,d,0,e,f,1)

Transforms act on the coordinates of primitives, so that a point with coordinates (x,y) is transformed to the point:

$$(a*x + c*y + e, b*x + d*y + f)$$

The transform can be used in any call following:

- GpiSetModelTransformMatrix
- GpiSetSegmentTransformMatrix
- GpiSetViewingTransformMatrix
- GpiSetDefaultViewMatrix.

Other similar helper functions are:

- GpiTranslate to apply a translation component
- GpiRotate to apply a rotation component.

Related Functions

- GpiRotate
- GpiTranslate
- GpiSetModelTransformMatrix
- GpiSetSegmentTransformMatrix
- GpiSetDefaultViewMatrix
- GpiSetViewingTransformMatrix

Example Code

In this example, the viewing transform matrix is scaled by a factor of 2.

```
#define INCL_GPITRANSFORMS
#include <OS2.H>

HPS hps;           /* presentation space handle */
MATRIXLF matlf;    /* transform matrix. */
POINTL ptlCenter;  /* center of rotation. */

GpiQueryViewingTransformMatrix(hps,
                                1L,
                                &matlf);

ptlCenter.x = 50L;
ptlCenter.y = 50L;

GpiRotate(hps,
           &matlf,
           TRANSFORM_REPLACE,
           MAKEFIXED(2,0), /* rotate 10 degrees left. the angle */
                               /* must be passed in fixed format. */
                               /* see the pmgpi.h file for a */
                               /* description of the MAKEFIXED macro. */
           &ptlCenter);
```


GpiSelectPalette —

Select Palette

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI or INCL_PM */
```

HPAL GpiSelectPalette (HPS hps, HPAL hpal)

This function selects a palette into a presentation space.

Parameters

- hps** (HPS) — input
Presentation-space handle.
- hpal** (HPAL) — input
Palette handle.
- NULLHANDLE** Set the color table for the presentation space to the default table (see GpiCreateLogColorTable).
- Other** Palette handle.

Returns

Old palette handle:

NULLHANDLE Successful completion, default (or loaded) color table was in effect

PAL_ERROR Error occurred

Otherwise Old palette handle.

Possible returns from WinGetLastError

- | | |
|----------------------------------|---|
| PMERR_INV_HPS | An invalid presentation-space handle was specified. |
| PMERR_PS_BUSY | An attempt was made to access the presentation space from more than one thread simultaneously. |
| PMERR_INV_HPAL | An invalid color palette handle was specified. |
| PMERR_INSUFFICIENT_MEMORY | The operation terminated through insufficient memory. |
| PMERR_PALETTE_BUSY | An attempt has been made to reset the owner of a palette when it was busy. |
| PMERR_INV_IN_AREA | An attempt was made to issue a function invalid inside an area bracket. This can be detected while the actual drawing mode is draw or draw-and-retain or during segment drawing or correlation functions. |

Remarks

This function overrides any color table previously loaded (see GpiCreateLogColorTable), or palette previously selected into this presentation space.

If *hpal* is specified as NULLHANDLE, then the color table for this presentation space is set to the default color table.

Palettes can be selected into more than one presentation space at a time, but only one palette can be selected into a given presentation space at any time.

If a palette is selected into a presentation space that is associated with a device context of type OD_MEMORY (see DevOpenDC), irreversible changes take place to any bit map selected into the device context.

GpiSelectPalette – Select Palette

If a palette is selected into a presentation space that is associated with a device context of type OD_METAFILE or OD_METAFILE_NOQUERY, the palette must apply to the entire picture, and must still be selected at the (last) time the metafile device context is dissociated from the presentation space.

Related Functions

- GpiAnimatePalette
- GpiCreatePalette
- GpiCreateLogColorTable
- GpiDeletePalette
- GpiQueryPalette
- GpiQueryPaletteInfo
- GpiSetPaletteEntries
- WinRealizePalette

Example Code

This function selects a palette into a presentation space.

```
#define INCL_GPILOGCOLORTABLE
#include <OS2.H>

HPS hps;          /* presentation-space handle. */
HPAL hpa101d, hpa1; /* old palette handle. */
hpa101d = GpiSelectPalette(hps, hpa1);
```

GpiSetArcParams — Set Arc Parameters

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

```
BOOL GpiSetArcParams (HPS hps, PARCPARAMS parcpArcParams)
```

This function sets the current arc parameters.

Parameters

hps (HPS) — input
Presentation-space handle.

parcpArcParams (PARCPARAMS) — input
Arc parameters.

This structure has four elements p, q, r, and s.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE An invalid coordinate value was specified.

Remarks

The arc parameters p, q, r, and s, define the shape and orientation of a ellipse that is used for subsequent GpiPointArc, GpiFullArc, and GpiPartialArc functions. For GpiFullArc and GpiPartialArc, they also determine the direction of drawing, as follows:

- If $p \cdot q > r \cdot s$ the direction is counterclockwise
- If $p \cdot q < r \cdot s$ the direction is clockwise
- If $p \cdot q = r \cdot s$ a straight line is drawn.

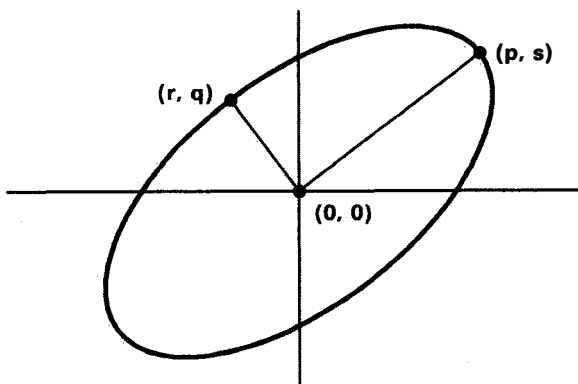


Figure 5-5. GpiSetArcParams Coordinate Points

For GpiFullArc and GpiPartialArc, these parameters also define the nominal size of the ellipse; this may be changed by using the multiplier.

GpiSetArcParams – Set Arc Parameters

For GpiPointArc, the size of the ellipse is determined by the three points specified on GpiPointArc.

The arc parameters define a transformation that maps the *unit circle* to the required ellipse, placed at the origin (0,0):

$$\begin{aligned}x' &= p*x + r*y \\ y' &= s*x + q*y\end{aligned}$$

With reference to Figure 5-5 on page 5-398, if $p*r + s*q = 0$, the transform is termed *orthogonal*, and the line from the origin (0,0) to the point (p,s) is either the radius of the circle, or half the major axis of the ellipse. The line from the origin to the point (r,q) is either the radius of the circle, or half of the minor axis of the ellipse.

For maximum accuracy, orthogonal transforms must be used. The matrix must not be singular.

The initial default values of arc parameters (unless changed with GpiSetDefArcParams) are:

$$\begin{array}{ll}p = 1 & r = 0 \\ s = 0 & q = 1\end{array}$$

producing a unit circle. (See Figure 5-6).

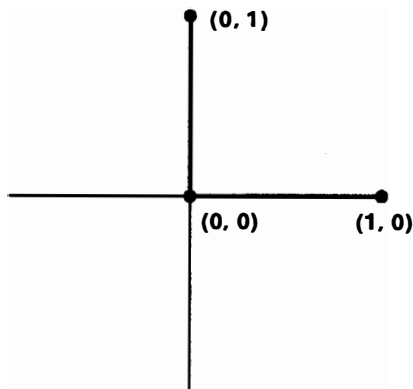


Figure 5-6. GpiSetArcParams Default Coordinates

Arc parameter transformation takes place in world coordinates. Any other non-square transformations in force change the shape of the figure accordingly.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the arc parameters is preserved.

Related Functions

- GpiQueryArcParams
- GpiSetDefArcParams
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix
- GpiSetLineType
- GpiSetLineWidth
- GpiFullArc
- GpiPartialArc
- GpiPointArc

GpiSetArcParams — Set Arc Parameters

Graphic Elements and Orders

Element Type: **OCODE_GSAP**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Arc Parameters**

Element Type: **OCODE_GPSAP**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Arc Parameters**

Example Code

This example uses the GpiSetArcParams function to draw an ellipse. The semimajor axis of the ellipse is 100, and the semiminor axis is 50. These values are in world coordinates, computed using the IP and IQ values of the arc parameters and the multiplier provided with the GpiFullArc function.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps; /* Presentation-space */
/* handle. */
ARCPARAMS arcp = { 4, 2, 0, 0 }; /* Arc parameters. */
/* This structure has four */
/* elements p, q, r, and s. */

POINTL pt1 = {100, 100};
GpiSetArcParams(hps, &arcp);
GpiMove(hps, &pt1);
GpiFullArc(hps, DRO_OUTLINE, MAKEFIXED(25, 0));
```

GpiSetAttrMode – Set Attribute Mode

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetAttrMode (HPS hps, LONG IMode)

This function specifies the current attribute mode.

Parameters

hps (HPS) – input
Presentation-space handle.

IMode (LONG) – input
Attribute mode:

AM_PRESERVE Preserve attributes
AM_NOPRESERVE Do not preserve attributes.

Returns

Success indicator:
TRUE Successful completion
FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_ATTR_MODE	An invalid mode parameter was specified with GpiSetAttrMode.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

Remarks

The attribute mode is used to specify whether a primitive attribute is to be preserved when set to a new value by a subsequent attribute setting call. The preserved value of an attribute can be restored using the GpiPop function. Any attributes that have been preserved in a called segment are automatically restored on return to the caller. The values of any attributes preserved in a *chained* segment, however, that are not restored using GpiPop by the end of the segment, are lost.

The following are affected:

- GpiSetArcParams
- GpiSetBackColor (applies individually by primitive type if GpiSetAttrs is used)
- GpiSetBackMix (applies individually by primitive type if GpiSetAttrs is used)
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear

GpiSetAttrMode – Set Attribute Mode

- GpiSetColor (applies individually by primitive type if GpiSetAttrs is used)
- GpiSetCurrentPosition
- GpiSetMix (applies individually by primitive type if GpiSetAttrs is used)
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom
- GpiSetMarkerBox
- GpiSetMarkerSet
- GpiSetMarker
- GpiSetModelTransformMatrix
- GpiSetPattern
- GpiSetPatternRefPoint
- GpiSetPatternSet
- GpiSetTag.

The initial value of the attribute mode, that is, its value before this function is issued, is **AM_NOPRESERVE**.

Attribute mode applies to attributes passed across the API using GpiSet... calls. What mode to use for a particular GpiSet... call is decided by the attribute mode current at the time the GpiSet... call is passed across the API. The mode may be changed at any time, and does not affect any attribute setting calls that have already been retained in the segment store.

Attribute mode only applies to individual GpiSet... calls (including GpiSetAttrs and calls such as GpiSetColor). It does not apply to any attribute setting calls passed across in bulk, such as GpiPutData, GpiElement, and GpiPlayMetaFile; these already indicate individually whether they should cause the attribute to be preserved.

Attribute mode cannot be set for GPIT_MICRO type presentation spaces, or for presentation spaces associated with OD_METAFILE_NOQUERY type device contexts (see GpiCreatePS and DevOpenDC). In these cases the presentation space behaves as if AM_NOPRESERVE is in operation.

Related Functions

- GpiPop
- GpiQueryAttrMode
- GpiResetPS
- GpiQueryAttrs
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetPS

GpiSetAttrMode — Set Attribute Mode

Example Code

This example uses the GpiSetAttrMode function to set the attribute mode to preserve.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;                                /* Presentation-space */
                                        /* handle. */

POINTL ptl[2] = { 50, 50, 100, 100, };

GpiSetColor(hps, CLR_BLUE);
GpiSetAttrMode(hps, AM_PRESERVE); /* sets attribute mode to */
                                   /* preserve. */
GpiSetColor(hps, CLR_GREEN);      /* changes color and saves old */
                                   /* color. */
GpiLine(hps, &ptl[0]);             /* draws green line */
GpiPop(hps, 1L);                  /* pops old attribute from */
                                   /* stack. */
GpiLine(hps, &ptl[1]);             /* draws blue line */
```


GpiSetAttrs — Set Attributes

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

```
BOOL GpiSetAttrs (HPS hps, LONG IPrimType, ULONG flAttrMask, ULONG flDefMask,  
                  PBUNDLE ppbunAttrs)
```

This function sets attributes for the specified primitive type.

Parameters

hps (HPS) — input
Presentation-space handle.

IPrimType (LONG) — input
Primitive type.

The primitive type for which attributes are to be set:

PRIM_LINE Line and arc primitives

PRIM_CHAR Character primitives

PRIM_MARKER Marker primitives

PRIM_AREA Area primitives

PRIM_IMAGE Image primitives.

flAttrMask (ULONG) — input
Attributes mask.

Each flag set indicates that either the corresponding flag in *flDefMask* is set, or the *ppbunAttrs* buffer contains data for the corresponding attribute. If all of the flags in *flAttrMask* are 0, the *ppbunAttrs* buffer address is not used.

Line attributes:

LBB_COLOR	Line color
LBB_MIX_MODE	Line mix
LBB_WIDTH	Line width
LBB_GEOM_WIDTH	Geometric line width
LBB_TYPE	Line type
LBB_END	Line end
LBB_JOIN	Line join.

Character attributes:

CBB_COLOR	Character color
CBB_BACK_COLOR	Character background color
CBB_MIX_MODE	Character mix
CBB_BACK_MIX_MODE	Character background mix
CBB_SET	Character set
CBB_MODE	Character mode
CBB_BOX	Character box
CBB_ANGLE	Character angle

GpiSetAttrs – Set Attributes

CBB_SHEAR	Character shear
CBB_DIRECTION	Character direction

Marker attributes:

MBB_COLOR	Marker color
MBB_BACK_COLOR	Marker background color
MBB_MIX_MODE	Marker mix
MBB_BACK_MIX_MODE	Marker background mix
MBB_SET	Marker set
MBB_SYMBOL	Marker symbol
MBB_BOX	Marker box.

Pattern attributes (areas):

ABB_COLOR	Area color
ABB_BACK_COLOR	Area background color
ABB_MIX_MODE	Area mix
ABB_BACK_MIX_MODE	Area background mix
ABB_SET	Pattern set
ABB_SYMBOL	Pattern symbol
ABB_REF_POINT	Pattern reference point.

Image attributes:

IBB_COLOR	Image color
IBB_BACK_COLOR	Image background color
IBB_MIX_MODE	Image mix
IBB_BACK_MIX_MODE	Image background mix.

flDefMask (ULONG) – input
Defaults mask.

Each flag set (and for which *flAttrMask* is also set) causes the corresponding attribute to be set to its default value.

ppbunAttrs (PBUNDLE) – input
Attributes.

This is a structure containing the attribute value of each attribute for which the *flAttrMask* flag is set (and which is not to be set to its default value), at the correct offset as specified below for the particular primitive type.

Primitive type:	Structure:
Line attributes	LINEBUNDLE
Character attributes	CHARBUNDLE
Marker attributes	MARKERBUNDLE
Pattern attributes (areas)	AREABUNDLE
Image attributes	IMAGEBUNDLE.

GpiSetAttrs – Set Attributes

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_PRIMITIVE_TYPE	An invalid primitive type parameter was specified with GpiSetAttrs or GpiQueryAttrs.
PMERR_UNSUPPORTED_ATTR	An unsupported attribute was specified in the attrmask with GpiSetAttrs or GpiQueryAttrs.
PMERR_INV_COLOR_ATTR	An invalid color attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_BACKGROUND_COL_ATTR	An invalid background color attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_MIX_ATTR	An invalid mix attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_LINE_WIDTH_ATTR	An invalid line width attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_GEOM_LINE_WIDTH_ATTR	An invalid geometric line width attribute value was specified.
PMERR_INV_LINE_TYPE_ATTR	An invalid line type attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_LINE_END_ATTR	An invalid line end attribute value was specified.
PMERR_INV_LINE_JOIN_ATTR	An invalid line join attribute value was specified.
PMERR_INV_CHAR_SET_ATTR	An invalid character setid attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_CHAR_MODE_ATTR	An invalid character mode attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_CHAR_DIRECTION_ATTR	An invalid character direction attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_CHAR_SHEAR_ATTR	An invalid character shear attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_CHAR_ANGLE_ATTR	The default character angle attribute value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

GpiSetAttrs – Set Attributes

PMERR_INV_MARKER_SET_ATTR	An invalid marker set attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_MARKER_SYMBOL_ATTR	An invalid marker symbol attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_PATTERN_SET_ATTR	An invalid pattern set attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_PATTERN_ATTR	An invalid pattern symbol attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.
PMERR_UNSUPPORTED_ATTR_VALUE	An attribute value was specified with GpiSetAttrs that is not supported.
PMERR_INV_PATTERN_SET_FONT	An attempt was made to use an unsuitable font as a pattern set.
PMERR_HUGE_FONTS_NOT_SUPPORTED	An attempt was made using GpiSetCharSet, GpiSetPatternSet, GpiSetMarkerSet, or GpiSetAttrs to select a font that is larger than the maximum size (64Kb) supported by the target device driver.

Remarks

Any attribute (for the specified primitive type) for which the appropriate flag is set in the *flAttrMask* has its value updated:

- If the corresponding flag in *flDefMask* is also set, the attribute is set to default.
- If the corresponding flag in *flDefMask* is not set, the attribute is set to the value specified in the *ppbunAttrs* structure.

Any attribute for which the appropriate flag in *flAttrMask* is not set is unchanged, regardless of the setting of the corresponding flag in *flDefMask*.

The *flDefMask* and *flAttrMask* parameters each contain flags: each attribute of the primitive type in question is represented by one flag.

The data in the *ppbunAttrs* buffer consists of a structure of attribute data. The layout of the structure is fixed for each primitive type. Only data for attributes for which the flag is set in *flAttrMask* (but not in *flDefMask*) is inspected; any other data is ignored.

Note: The buffer need be no longer than is necessary to contain the data for the highest offset attribute referenced.

If default values of attributes are required, they must be requested using the *flDefMask*. The system does not recognize attribute values whose meaning would normally be "use default" in the *ppbunAttrs* buffer.

Where possible, invalid color values are detected by this call and cause an error (PMERR_INV_COLOR_ATTR or PMERR_INV_BACKGROUND_COL_ATTR) to be logged. Some invalid color values cannot be detected until draw time at which point the implementation optionally defaults them, or causes the above error to be logged. If an attempt to set an invalid value by this function is detected, none of the specified attributes is changed. Note, however, that some invalid attribute values (for example, colors and mixes) may not be detected until the attribute is used.

GpiSetAttrs — Set Attributes

If this function occurs within a path bracket, it must only set:

- Line attributes, other than the geometric line width
- Character attributes, other than foreground or background colors or mixes
- Marker attributes, other than foreground or background colors or mixes.

The default values of attributes can be changed with GpiSetDefAttrs.

The attribute mode (see GpiSetAttrMode) determines whether the current values of the attributes are preserved. If they are, one “push” call is generated for each affected attribute, in the order in which the attributes are specified, in the appropriate xxxBUNDLE structure.

Related Functions

- GpiPop
- GpiQueryAttrs
- GpiSetAttrMode
- GpiSetDefAttrs

Graphic Elements and Orders

The element type depends on the *IPrimType* parameter.

For each element, the “Set” orders are generated, if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE, and the “Push and Set” orders if it is AM_PRESERVE. In either instance, a particular order is generated only if the corresponding attribute is being set with this function, as specified on the *ppbunAttrs* parameter.

Element Type: ETYPE_LINEBUNDLE
Generated if *IPrimType* is PRIM_LINE.

Order: Set Individual Attribute

One of these for each of LBB_COLOR, and LBB_MIX_MODE, as required.

Order: Set Fractional Line Width

LBB_WIDTH, as required.

Order: Set Stroke Line Width

LBB_GEOM_WIDTH, as required.

Order: Set Line Type

LBB_TYPE, as required.

Order: Set Line End

LBB_END, as required.

Order: Set Line Join

LBB_JOIN, as required.

GpiSetAttrs – Set Attributes

As many as required of the following are generated if the attribute mode is AM_PRESERVE:

Order: Push and Set Individual Attribute

One of these for each of LBB_COLOR, and LBB_MIX_MODE, as required.

Order: Push and Set Fractional Line Width

LBB_WIDTH, as required.

Order: Push and Set Stroke Line Width

LBB_GEOM_WIDTH, as required.

Order: Push and Set Line Type

LBB_TYPE, as required.

Order: Push and Set Line End

LBB_END, as required.

Order: Push and Set Line Join

LBB_JOIN, as required.

Element Type: ETYPE_CHARBUNDLE

Generated if *IPrimType* is PRIM_CHAR.

Order: Set Individual Attribute

One of these for each of CBB_COLOR, CBB_BACK_COLOR, CBB_MIX_MODE, and CBB_BACK_MIX_MODE, as required.

Order: Set Character Set

CBB_SET

Order: Set Character Precision

CBB_MODE

Order: Set Character Cell

CBB_BOX

Order: Set Character Angle

CBB_ANGLE

Order: Set Character Shear

CBB_SHEAR

Order: Set Character Direction

CBB_DIRECTION

As many as required of the following are generated if the attribute mode is AM_PRESERVE:

Order: Push and Set Individual Attribute

One of these for each of CBB_COLOR, CBB_BACK_COLOR, CBB_MIX_MODE, and CBB_BACK_MIX_MODE, as required.

Order: Push and Set Character Set

CBB_SET

Order: Push and Set Character Precision

CBB_MODE

Order: Push and Set Character Cell

CBB_BOX

Order: Push and Set Character Angle

CBB_ANGLE

Order: Push and Set Character Shear

CBB_SHEAR

Order: Push and Set Character Direction

CBB_DIRECTION

GpiSetAttrs – Set Attributes

Element Type: **ETYPE_MARKERBUNDLE**

Generated if *IPrimType* is PRIM_MARKER.

Order: **Set Individual Attribute**

One of these for each of MBB_COLOR, MBB_BACK_COLOR, MBB_MIX_MODE, and MBB_BACK_MIX_MODE, as required.

Order: **Set Marker Set**

MBB_SET

Order: **Set Marker Symbol**

MBB_SYMBOL

Order: **Set Marker Cell**

MBB_BOX

As many as required of the following are generated if the attribute mode is AM_PRESERVE:

Order: **Push and Set Individual Attribute**

One of these for each of MBB_COLOR, MBB_BACK_COLOR, MBB_MIX_MODE, and MBB_BACK_MIX_MODE, as required.

Order: **Push and Set Marker Set**

MBB_SET

Order: **Push and Set Marker Symbol**

MBB_SYMBOL

Order: **Push and Set Marker Cell**

MBB_BOX

Element Type: **ETYPE_AREABUNDLE**

Generated if *IPrimType* is PRIM_AREA.

Order: **Set Individual Attribute**

One of these for each of ABB_COLOR, ABB_BACK_COLOR, ABB_MIX_MODE, and ABB_BACK_MIX_MODE, as required.

Order: **Set Pattern Set**

ABB_SET

Order: **Set Pattern Symbol**

ABB_SYMBOL

Order: **Set Pattern Reference Point**

ABB_REF_POINT

As many as required of the following are generated if the attribute mode is AM_PRESERVE:

Order: **Push and Set Individual Attribute**

One of these for each of ABB_COLOR, ABB_BACK_COLOR, ABB_MIX_MODE, and ABB_BACK_MIX_MODE, as required.

Order: **Push and Set Pattern Set**

ABB_SET

Order: **Push and Set Pattern Symbol**

ABB_SYMBOL

Order: **Push and Set Pattern Reference Point**

ABB_REF_POINT

Element Type: **ETYPE_IMAGEBUNDLE**

Generated if *IPrimType* is PRIM_IMAGE.

Order: **Set Individual Attribute**

One of these for each of IBB_COLOR, IBB_BACK_COLOR, IBB_MIX_MODE, and IBB_BACK_MIX_MODE, as required.

GpiSetAttrs — Set Attributes

As many as required of the following are generated if the attribute mode is AM_PRESERVE:

Order: Push and Set Individual Attribute

One of these for each of IBB_COLOR, IBB_BACK_COLOR, IBB_MIX_MODE, and IBB_BACK_MIX_MODE, as required.

Example Code

This example uses the GpiSetAttrs function to set the line color to red and the line width to its default value.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;                                /* Presentation-space */
                                        /* handle.             */

LINEBUNDLE lband;
lband.lColor = CLR_RED;

GpiSetAttrs(hps,                        /* presentation-space handle */
            PRIM_LINE,                  /* line primitive.           */
            LBB_COLOR | LBB_WIDTH,     /* sets line color and width. */
            LBB_WIDTH,                 /* sets line width to default */
            &lband);                   /* buffer for attributes.     */
```


GpiSetBackColor – Set Background Color

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetBackColor (HPS hps, LONG IColor)

This function sets the current background color index attribute, for each individual primitive type, to the specified value.

Parameters

hps (HPS) – input
Presentation-space handle.

IColor (LONG) – input
Background color:

For a loadable color table, values 0 through n correspond to the color index (or RGB) values.

CLR_FALSE All color planes are zeros.

CLR_TRUE All color planes are ones.

CLR_DEFAULT Set to default value. This is the natural background color for the device. For a display, it is the default window color (SYSCLR_WINDOW: see WinSetSysColors). For a printer, it is the paper color. The default can be changed by setting new system colors from the control panel for the display, or by selecting a paper color for a printer (if allowed by the device driver), or set explicitly with GpiSetDefAttrs.

CLR_WHITE White (default color table, or index=RGB loaded color table). For a loaded, realized, color table, it is the nearest available color to white.

CLR_BLACK Black (default color table, or index=RGB loaded color table). For a loaded, realized, color table, it is the nearest available color to black.

CLR_BACKGROUND Reset color, used by GpiErase. This is the natural background color for the device. For a display, it is the default window color (SYSCLR_WINDOW: see WinSetSysColors) for the default color table. For a printer, it is the paper color. For a loaded color table, it is color index 0. For an RGB color table, it is color 000000 (black).

CLR_BLUE Blue (default color table).

CLR_RED Red (default color table).

CLR_PINK Pink (default color table).

CLR_GREEN Green (default color table).

CLR_CYAN Cyan (default color table).

CLR_YELLOW Yellow (default color table).

CLR_NEUTRAL Neutral (default color table). A device-dependent color, that for the default color table provides a contrasting color to CLR_BACKGROUND. For a display, it is the default window text color (SYSCLR_WINDOWTEXT: see WinSetSysColors). For a printer, it is a color that contrasts with the paper color. For a loaded color table, it is color index 7; in RGB mode it is interpreted as color 000007.

CLR_DARKGRAY Dark gray (default color table).

CLR_DARKBLUE Dark blue (default color table).

CLR_DARKRED Dark red (default color table).

GpiSetBackColor – Set Background Color

CLR_DARKPINK	Dark pink (default color table).
CLR_DARKGREEN	Dark green (default color table).
CLR_DARKCYAN	Dark cyan (default color table).
CLR_BROWN	Brown (default color table).
CLR_PALEGRAY	Pale gray (default color table).

For a loadable color table, values 0 through n correspond to the color index (or RGB) values.

Returns

Success indicator:

TRUE	Successful completion
FALSE	Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_BACKGROUND_COL_ATTR	An invalid background color attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

Remarks

Note that if the background mix is BM_LEAVEALONE (the default setting), the background color is not seen.

An attempt to set a negative color value, other than one for which a constant is defined, causes the error PMERR_INV_COLOR_ATTR to be logged. Other color values are allowed, although an error is generated when the color value is needed for drawing if it is invalid for the color table in use at that time (see GpiCreateLogColorTable).

For details of how colors are handled on monochrome devices, also see GpiCreateLogColorTable.

The attribute mode determines whether the current value of the background color attribute is preserved. If it is, the values of the background color attribute, for each primitive type, are preserved, and a single GpiPop function restores them.

This function must not be used within a path or area bracket.

Related Functions

- GpiQueryBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt

GpiSetBackColor — Set Background Color

- GpiBox
- GpiMarker
- GpiPolyMarker
- GpiFullArc
- GpiPartialArc
- GpiPointArc
- GpiPolyFillet
- GpiPolyFilletSharp
- GpiPolySpline
- GpiBeginArea
- GpiEndArea

Graphic Elements and Orders

Element Type: **OCODE_GSBICOL**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Background Indexed Color**

Element Type: **OCODE_GPSBICOL**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Background Indexed Color**

Example Code

This is an example of a function used to repaint the window when a WM_PAINT message is issued.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

void ClientPaint( HWND hwnd )
{
    POINTL pt;
    HPS hps; /* Presentation space handle */
    RECTL rcl; /* Window rectangle */
    long clrText;

    /* Obtain a cache PS and set color
       and background mix attributes */
    hps = WinBeginPaint( hwnd, (HPS)NULLHANDLE, (PRECTL)&rcl );
    GpiSetColor( hps, clrText );
    GpiSetBackColor( hps, CLR_BACKGROUND ); /* set background
                                             to white. */
    GpiSetBackMix( hps, BM_OVERPAINT );
}
```

GpiSetBackMix – Set Background Mix

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetBackMix (HPS hps, LONG IMixMode)

This function sets the current background mix attribute for each individual primitive type.

Parameters

hps (HPS) – input
Presentation-space handle.

IMixMode (LONG) – input
Background-mix mode.

Background mixes marked with an asterisk (*) are mandatory for all devices.

The currently associated device supports any of the mixes specified as supported in DevQueryCaps (CAPS_BACKGROUND_MIX_SUPPORT). Any other valid mixes can be supported for some primitive types but otherwise result in BM_LEAVEALONE. An error is raised only if the value specified is not one of those listed.

For more information on mixing, see GpiSetMix.

BM_DEFAULT The default value (BM_LEAVEALONE unless changed with GpiSetDefAttrs).

BM_OR Logical-OR.

BM_OVERPAINT The background of the primitive takes precedence over whatever is underneath. (*)

BM_XOR Exclusive-OR.

BM_LEAVEALONE The background of the primitive has no effect on what is underneath. (*)

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_BACKGROUND_COL_ATTR An invalid background color attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

Remarks

The background mix attribute controls the way that the background color of a primitive is combined with the color of any primitive that it overlaps.

These primitives are affected by the background mix attribute:

Areas The background of an area is defined to be every pel within the area that is not set by the shading pattern.

Text The background of a character is the complete character box.

Images For an image, the background is every pel within the image that is not set.

GpiSetBackMix — Set Background Mix

Markers The background of a marker is the complete marker box.

Note: When the background mix is BM_LEAVEALONE (initial default) the background color is not seen.

The attribute mode determines whether the current value of the background mix attribute is preserved. If it is, the values of the background mix attribute for each primitive type are preserved, and a single GpiPop function restores them.

This function should not be used within a path or area bracket.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see “Metafile Restrictions” on page G-1.

Related Functions

- GpiQueryBackMix
- GpiSetBackColor
- GpiSetColor
- GpiSetMix
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt
- GpiBox
- GpiMarker
- GpiPolyMarker
- GpiFullArc
- GpiBeginArea
- GpiEndArea

Graphic Elements and Orders

Element Type: **OCODE_GSBMX**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Background Mix**

Element Type: **OCODE_GPSBMX**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Background Mix**

GpiSetBackMix — Set Background Mix

Example Code

This is an example of a function used to repaint the window when a WM_PAINT message is issued.

```
VOID cdecl ClientPaint( HWND hwnd )
{
    POINTL pt;
    HPS hps; /* Presentation space handle */
    RECTL rcl; /* Window rectangle */

    /* Obtain a cache PS and set color
       and background mix attributes */
    hps = WinBeginPaint( hwnd, (HPS)NULLHANDLE, (PRECTL)&rcl );
    GpiSetColor( hps, clrText );
    GpiSetBackColor( hps, CLR_BACKGROUND ); /* set background
                                              to white. */
    GpiSetBackMix( hps, BM_OVERPAINT );
    /* the background of the primitive takes
       over whatever is underneath. */
}
```

GpiSetBitmap — Set Bit Map

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

HBITMAP GpiSetBitmap (HPS hps, HBITMAP hbm)

This function sets a bit map as the currently selected bit map in a memory device context.

Parameters

hps (HPS) — input
Presentation-space handle.

hbm (HBITMAP) — input
Handle of the bit map to be set.

A null handle causes the currently selected bit map, in the associated device, to be freed.

It is an error if the bit map is currently tagged for area shading (see GpiSetBitmapId).

Returns

Old bit-map handle:

NULLHANDLE Correct (null handle)

HBM_ERROR Error

Otherwise Old bit-map handle.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HBITMAP An invalid bit-map handle was specified.

PMERR_BITMAP_IN_USE An attempt was made either to set a bit map into a device context using GpiSetBitmap while it was already selected into an existing device context, or to tag a bit map with a local pattern set identifier (setid) using GpiSetBitmapId while it was already tagged with an existing setid.

PMERR_INCOMPATIBLE_BITMAP An attempt was made to select a bit map or perform a BitBlt operation on a device context that was incompatible with the format of the bit map.

PMERR_HBITMAP_BUSY An internal bit map busy error was detected. The bit map was locked by one thread during an attempt to access it from another thread.

Remarks

The specified presentation space must be currently associated with a memory device context. The device context can represent a different physical device from the one that the bit map originally loaded or created, providing its format is convertible to one supported on the new device. This is ensured when one of the standard formats is being used.

If a bit map is already current in the device context, the handle of this bit map is returned, before the new bit map is selected.

It is an error if the new bit map is already selected as the current bit map in any device context.

Related Functions

- GpiBitBlt
- GpiCreateBitmap
- GpiDeleteBitmap
- GpiDrawBits
- GpiLoadBitmap
- GpiQueryBitmapBits
- GpiQueryBitmapDimension
- GpiQueryBitmapHandle
- GpiQueryBitmapParameters
- GpiQueryDeviceBitmapFormats
- GpiSetBitmapBits
- GpiSetBitmapDimension
- GpiSetBitmapId
- GpiWCBitBlt
- WinDrawBitmap
- WinGetSysBitmap

Example Code

This example uses the GpiSetBitmap function to set a newly created bit map in the device context for the associated presentation space. Once set, the example initializes the bit map image by drawing in the presentation space.

```
#define INCL_GPIBITMAPS
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS      hps;                                /* Presentation space handle */
BITMAPINFOHEADER2 bmp = {12, 64, 64, 1, 1}; /* 64x64 mono bit map */
HBITMAP hbm, hbmOld;
POINTL ptlStart = { 0, 0 };
POINTL aptlTriangle[3] = { 32, 32, 63, 63, 0, 0 };
POINTL ptl = { 63, 63 };

hbm = GpiCreateBitmap(hps,
                      &bmp,
                      0L,
                      NULL,
                      NULL);

/* Set the bit map and draw in it. */

/* sets bit map in device context */
hbmOld = GpiSetBitmap(hps, hbm);
GpiMove(hps, &ptlStart);
GpiBox(hps, DRO_FILL, &ptl, 0L, 0L); /* fills in the bit map */
GpiPolyLine(hps,                                /* draws a triangle */
            1L,
            aptlTriangle);
GpiSetBitmap(hps, hbmOld);                      /* restores the old bit map */
```


GpiSetBitmapBits – Set Bit-Map Bits

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI or INCL_PM */
```

```
LONG GpiSetBitmapBits (HPS hps, LONG IScanStart, LONG IScans, PBYTE pbBuffer,  
PBITMAPINFO2 pbmi2InfoTable)
```

This function transfers bit-map data from application storage to a bit map.

Parameters

- hps** (HPS) – input
Presentation-space handle.
- IScanStart** (LONG) – input
Line number
Scan-line number at which the data transfer is to start, counting from 0 as the bottom line.
- IScans** (LONG) – input
Number of scan lines to be transmitted.
- pbBuffer** (PBYTE) – input
Bit-map data buffer.
Address in application storage from which the bit-map data is to be copied.
- pbmi2InfoTable** (PBITMAPINFO2) – input
Bit-map information table.

Returns

- Number of scan lines actually set:
- ≥ 0 Number of scan lines actually set
- GPI_ALTERERROR** Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_INV_INFO_TABLE	An invalid bit-map info table was specified with a bit-map operation.
PMERR_NO_BITMAP_SELECTED	An attempt has been made to operate on a memory device context that has no bit map selected.
PMERR_INV_SCAN_START	An invalid scanstart parameter was specified with a bitmap function.
PMERR_INCORRECT_DC_TYPE	An attempt was made to perform a bit-map operation on a presentation space associated with a device context of a type that is unable to support bit-map operations.

GpiSetBitmapBits — Set Bit-Map Bits

Remarks

The presentation space must be currently associated with a memory device context that has a bit map currently selected.

Note: This function does not set bits directly to any other kind of device.

If the format of the supplied bit map does not match that of the device, it is converted, using the supplied bit-map information table. The standard bit-map formats are supported, plus any known to be supported by the device; see GpiQueryDeviceBitmapFormats.

Related Functions

- GpiBitBit
- GpiCreateBitmap
- GpiDeleteBitmap
- GpiDrawBits
- GpiLoadBitmap
- GpiQueryBitmapBits
- GpiQueryBitmapDimension
- GpiQueryBitmapHandle
- GpiQueryBitmapParameters
- GpiQueryDeviceBitmapFormats
- GpiSetBitmap
- GpiSetBitmapDimension
- GpiSetBitmapId
- GpiWCBitBit
- WinDrawBitmap
- WinGetSysBitmap

GpiSetBitmapBits —

Set Bit-Map Bits

Example Code

This example uses the GpiSetBitmapBits function to copy image data to a bit map in a presentation space associated with a memory device context.

```
#define INCL_GPIPRIMITIVES
#define INCL_GPIBITMAPS
#define INCL_DOSMEMMGR
#define INCL_WINDIALOGS
#include <OS2.H>

HPS hps; /* Presentation space handle */
BITMAPINFOHEADER2 bmp = { 12, 32, 16, 1, 1 };
SEL sel;
PBITMAPINFOHEADER2 pbmi;
BYTE pbBuffer[16]; /* buffer for image data */
ULONG cbBitmapInfo;
HBITMAP hbm, hbmOld;
BOOL f;
/* Allocate space for the bit-map information table. */

cbBitmapInfo = sizeof(BITMAPINFO) + sizeof(RGB);
f = (BOOL)DosAllocMem((PPVOID)pbmi,
    (ULONG)cbBitmapInfo,
    PAG_READ |
    PAG_WRITE |
    PAG_COMMIT);
if (f) {
    WinMessageBox(HWND_DESKTOP, HWND_DESKTOP,
        "Sorry, Not enough memory",
        NULL,
        0,
        MB_OK);
    return;
}

/* Initialize the bit-map information table. */

pbmi->cbFix = 12;
pbmi->cx = 32;
pbmi->cy = 16;
pbmi->cPlanes = 1;
pbmi->cBitCount = 1;

/* Create the bit map, set to the device,
and copy the image data. */

hbm = GpiCreateBitmap(hps,
    pbmi,
    0L,
    NULL,
    NULL);
hbmOld = GpiSetBitmap(hps, hbm);
GpiSetBitmapBits(hps,
    0L,
    (LONG)bmp.cy,
    pbBuffer,
    pbmi);
GpiSetBitmap(hps, hbmOld);
```

GpiSetBitmapDimension – Set Bit-Map Dimension

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetBitmapDimension (HBITMAP hbm, PSIZEL pszlBitmapDimension)

This function associates a width and height with a bit map, in units of 0.1 millimeter.

Parameters

hbm (HBITMAP) – input
Bit-map handle.

pszlBitmapDimension (PSIZEL) – input
Width and height of bit map.

The width and height, respectively, of the bit map in units of 0.1 millimeter.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HBITMAP

An invalid bit-map handle was specified.

PMERR_HBITMAP_BUSY

An internal bit map busy error was detected. The bit map was locked by one thread during an attempt to access it from another thread.

Remarks

The values set are not used internally by the system, but are retained with the bit map and can be retrieved with GpiQueryBitmapDimension.

Related Functions

- GpiBitBit
- GpiCreateBitmap
- GpiDeleteBitmap
- GpiDrawBits
- GpiLoadBitmap
- GpiQueryBitmapBits
- GpiQueryBitmapDimension
- GpiQueryBitmapHandle
- GpiQueryBitmapParameters
- GpiQueryDeviceBitmapFormats
- GpiSetBitmap
- GpiSetBitmapBits
- GpiSetBitmapId
- GpiWCBitBit
- WinDrawBitmap
- WinGetSysBitmap

GpiSetBitmapDimension — Set Bit-Map Dimension

Example Code

This example uses the GpiSetBitmap and GpiSetBitmapDimension function to set a newly created bit map in the device context for the associated presentation space. Once set, the example initializes the bit-map image by drawing in the presentation space.

```
#define INCL_GPIBITMAPS
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS      hps;                /* Presentation space handle */
BOOL     fSuccess;           /* Success indicator */
BITMAPINFOHEADER2 bmp = {12, 64, 64, 1, 1}; /* 64x64 mono bit map */
HBITMAP  hbm, hbmOld;
POINTL   ptlStart = { 0, 0 };
POINTL   aptlTriangle[3] = { 32, 32, 63, 63, 0, 0 };
POINTL   ptl = { 63, 63 };
SIZEL    sizlBitmapDimension = {100, 100};
        /* The width and height, */
        /* respectively, of the bit */
        /* map in units of 0.1 */
        /* millimeter. */

hbm = GpiCreateBitmap(hps,
                    &bmp,
                    0L,
                    NULL,
                    NULL);

/* Set the bit map and draw in it. */

fSuccess = GpiSetBitmapDimension(hbm,
                                &sizlBitmapDimension);
hbmOld = GpiSetBitmap(hps, hbm); /* sets bit map
                                in device context */

GpiMove(hps, &ptlStart);
GpiBox(hps, DRO_FILL, &ptl, 0L, 0L); /* fills in the bit map */
GpiPolyLine(hps, /* draws a triangle */
            1L,
            aptlTriangle);
GpiSetBitmap(hps, hbmOld); /* restores the old bit map*/
```

GpiSetBitmapId – Set Bit-Map Identifier

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetBitmapId (HPS hps, HBITMAP hbm, LONG lCld)

This function tags a bit map with a local identifier, so that it can be used as a pattern set, containing a single member.

Parameters

hps (HPS) – input

Presentation-space handle.

hbm (HBITMAP) – input

Bit-map handle.

The bit map must not be currently selected into a device context (see GpiSetBitmap).

lCld (LONG) – input

Local identifier with which the bit map is to be tagged.

Valid values are in the range 1 through 254.

It is an error if the local identifier is already used to refer to a font or bit map.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from GetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HBITMAP

An invalid bit-map handle was specified.

PMERR_INV_SETID

An invalid setid parameter was specified.

PMERR_SETID_IN_USE

An attempt was made to specify a setid that was already in use as the currently selected character, marker or pattern set.

PMERR_BITMAP_IN_USE

An attempt was made either to set a bit map into a device context using GpiSetBitmap while it was already selected into an existing device context, or to tag a bit map with a local pattern set identifier (setid) using GpiSetBitmapId while it was already tagged with an existing setid.

PMERR_HBITMAP_BUSY

An internal bit map busy error was detected. The bit map was locked by one thread during an attempt to access it from another thread.

GpiSetBitmapId – Set Bit-Map Identifier

Remarks

To use the bit map for area shading (or as the pattern in a GpiBitBlt or GpiWCBitBlt operation), a GpiSetPatternSet must be issued with the specified local identifier.

Any bit map of a format supported by the device can be specified. However, it may be simplified before use (see GpiSetPatternSet).

GpiDeleteSetId can subsequently be used to release the tag.

Related Functions

- GpiBitBlt
- GpiCreateBitmap
- GpiDeleteBitmap
- GpiDrawBits
- GpiLoadBitmap
- GpiQueryBitmapBits
- GpiQueryBitmapDimension
- GpiQueryBitmapHandle
- GpiQueryBitmapParameters
- GpiQueryDeviceBitmapFormats
- GpiSetBitmap
- GpiSetBitmapBits
- GpiSetBitmapDimension
- GpiWCBitBlt
- WinDrawBitmap
- WinGetSysBitmap
- GpiDeleteSetId
- GpiSetPatternSet

Example Code

This function tags a bit map with a local identifier, so that it can be used as a pattern set, containing a single member.

```
#define INCL_GPIBITMAPS
#include <OS2.H>

HPS      hps;    /* Presentation space handle */
HBITMAP  hbm;    /* bit-map handle. */
LONG  lid = 23;  /* local identifier. */

GpiSetBitmapId(hps,
               hbm,
               lid);
```

GpiSetCharAngle – Set Character Angle

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetCharAngle (HPS hps, PGRADIENTL pgradAngle)

This function specifies the angle of the baseline for the characters in a string, as a relative vector.

Parameters

hps (HPS) – input
Presentation-space handle.

pgradAngle (PGRADIENTL) – input
Baseline angle.

The baseline angle is defined in terms of the relative coordinates of the point *pgradAngle* (*x*, *y*).

If both *x* and *y* are 0, the character angle is reset to the default value. This default value is (1,0), unless changed with GpiSetDefAttrs.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

Remarks

The coordinates of the point *pgradAngle* specify integer values for the coordinates of the end of a line starting at the origin (0,0); the base line for subsequent character strings is parallel to this line.

The effect of the baseline angle attribute depends on the value of the character mode attribute (see GpiSetCharMode), and whether the current font is an outline or a raster font, as described below.

When the character mode is set to CM_MODE1, and the current font is a raster font, the character angle can be ignored.

When the character mode is set to CM_MODE2, and the current font is a raster font, the angle is used to determine the position of each character, but the orientations of characters within the character box may not be affected by changes in character angle. If this is so, the characters are positioned so that the lower left-hand corners of the character definitions are placed at the lower left-hand corners of the character boxes after all transforms have been applied. This is illustrated in Figure 5-7 on page 5-428.

GpiSetCharAngle – Set Character Angle

For illustrative purposes, the figure shows all character reference points at their bottom left-hand corner.

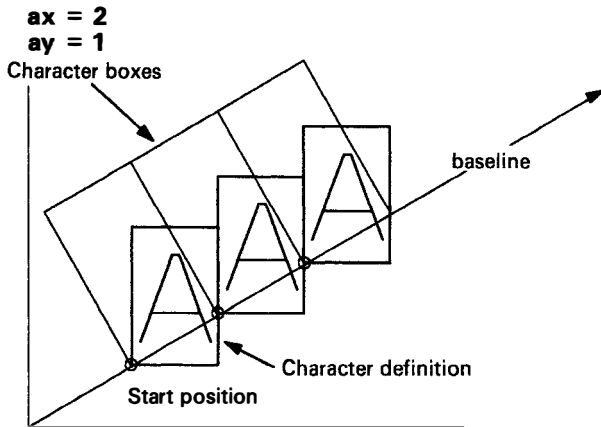


Figure 5-7. Character Angle and Mode-2 Text Positioning

When the character mode is set to CM_MODE3, or when the current font is an outline font, the angle is observed accurately, and the character boxes are rotated to be normal (perpendicular) to the character baseline. If the world coordinate system is such that one x-axis unit is not physically equal to one y-axis unit, a rotated character string appears to be sheared.

This function must not be issued in an area bracket.

The attribute mode determines whether the current value of the baseline angle attribute is preserved.

Related Functions

- GpiQueryCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix
- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt

Graphic Elements and Orders

Element Type: **OCODE_GSCA**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Character Angle**

Element Type: **OCODE_GPSCA**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Character Angle**

Example Code

This function resets the angle of the baseline for the characters in a string, as a relative vector.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;
GRADIENL gradlAngle = {0L, 0L};
GpiSetCharAngle(hps,
                &gradlAngle);
```

GpiSetCharBox – Set Character Box

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetCharBox (HPS hps, PSIZEF pszfxBox)

This function sets the current character-box attribute to the specified value.

Parameters

hps (HPS) – input
Presentation-space handle.

pszfxBox (PSIZEF) – input
Character-box size in world coordinates.

The width determines the spacing of consecutive characters along the baseline.

Both width and height can be positive, negative, or zero.

When either parameter is negative, the spacing occurs in the opposite direction to normal and each character is drawn reflected in character-mode 3. Thus, for example, a negative height in the standard direction in mode 3 means that the characters are drawn upside down, and the string drawn below the baseline (assuming no other transformations cause inversion).

A zero character width or height is also valid; here, the string of characters becomes a line. If both are zero, the string is drawn as a single point.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

Remarks

The parameter *pszfxBox* specifies values for the width and height of a character box in world coordinates.

Whether these values are used when character strings are drawn depends on the type of font being used (raster or outline), and on the value of the character mode attribute (see the *GpiSetCharMode* function).

For raster fonts, where the character box is used only for positioning in character mode *CM_MODE2*, the character box width corresponds to the *lEmInc* font metric (see *FONTMETRICS*). For proportionally-spaced raster fonts, the effective spacing for a given character is the character box width, scaled by the ratio of that character's width to *lEmInc*.

For outline fonts, characters are defined in font definition space. The *sXDeviceRes* and *sYDeviceRes* fields (see *FONTMETRICS*) define a rectangle in font definition space that is mapped to the character box rectangle (modified by the character angle and shear attributes) in world coordinates. *sYDeviceRes* corresponds to the font point size in font definition space, and therefore the character box height corresponds to the font point size in world coordinates. This is typically less than the *lMaxBaselineExt*.

GpiSetCharBox – Set Character Box

The effective width of each character from an outline font is the character box width, scaled by the ratio of the width of the character to *sXDeviceRes*. The *IAveCharWidth* (for a proportionally-spaced font) will therefore typically be smaller than the character box width. Indeed, because of differences in font design, *IAveCharWidth* and *IMaxBaselineExt* vary between different fonts, even when the character box dimensions are the same.

IEmlnc and *IEmHeight* are always equal to the character box width and height, respectively.

To cause characters of a given point-size to be generated using an outline font, establish a world coordinate space with specific metrics (for example, specify PU_TWIPS on *GpiCreatePS*), and set the character box height to the required point size. Because *sXDeviceRes* and *sYDeviceRes* are normally equal, the character box width should also be set equal to the height, if characters are required with the same aspect ratio as defined in the font (assuming that world coordinate space is *isotropic*).

The initial default value of the character box is the device-coordinates value returned by *DevQueryCaps* (CAPS_GRAPHICS_CHAR_WIDTH and CAPS_GRAPHICS_CHAR_HEIGHT), for the currently associated device, converted to page coordinates.

Note: In general the initial default value is rectangular, and to avoid character distortion, the character box should normally be set explicitly to be square if an outline font might be used (assuming that world coordinate space is *isotropic*).

The default value can be changed with *GpiSetDefAttrs*.

GpiSetCharBox must not be issued in an area bracket.

The attribute mode (see *GpiSetAttrMode*) determines whether the current value of the character-box size attribute is preserved.

Related Functions

- *GpiQueryCharBox*
- *GpiSetCharAngle*
- *GpiSetCharDirection*
- *GpiSetCharMode*
- *GpiSetCharSet*
- *GpiSetCharShear*
- *GpiPop*
- *GpiSetAttrMode*
- *GpiSetAttrs*
- *GpiSetDefAttrs*
- *GpiSetBackColor*
- *GpiSetBackMix*
- *GpiSetColor*
- *GpiSetMix*
- *GpiCharString*
- *GpiCharStringAt*
- *GpiCharStringPos*
- *GpiCharStringPosAt*
- *GpiQueryCharStringPos*
- *GpiQueryCharStringPosAt*

GpiSetCharBox — Set Character Box

Graphic Elements and Orders

Element Type: **OCODE_GSCC**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE. Note that GpiCreateLogFont can also generate this element type.

Order: **Set Character Cell**

Element Type: **OCODE_GPSCC**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Character Cell**

Example Code

This function sets the current character-box attribute to the specified value.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
```

```
HPS    hps;    /* Presentation space handle */
SIZEF sizfCharBox; /* Character-box size in */
                /* world coordinates.    */

sizfCharBox.cx = 8L<<16; /* values are shifted to the */
sizfCharBox.cy = 20L<<16; /* to make them fixed.    */
GpiSetCharBox( hps,
               &sizfCharBox );
```

GpiSetCharBreakExtra – Set Character Break Extra

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetCharBreakExtra (HPS hps, FIXED fxBreakExtra)

This function specifies an extra increment to be used for spacing break characters in a string.

Parameters

hps (HPS) – input

Presentation-space handle.

fxBreakExtra (FIXED) – input

Character-break-extra value.

The value can be negative, 0, or positive:

- A negative value reduces the effective width of break characters.
- A value of 0 results in normal spacing.
- A positive value increases the effective width of break characters.

The value is in world coordinates.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

Remarks

The character-break-extra attribute provides a spacing value that increases or decreases the spacing for break characters in a string. The break character is defined by the font, and can be found by calling GpiQueryFonts (*sBreakChar* field in the FONTMETRICS structure).

The break-extra spacing is additional to the spacing generated for other reasons, for example:

- The spacing determined by the font, including proportional spacing and kerning, if applicable
- The vector of increment values, see GpiCharStringPos, GpiCharStringPosAt, GpiQueryCharStringPos and GpiQueryCharStringPosAt
- Extra spacing, see GpiSetCharExtra.

Break-extra spacing applies to character strings either within or outside a path definition (see GpiBeginPath). The effect of the character-break-extra attribute applies whatever the value of the character-mode attribute (see GpiSetCharMode), and for both outline and *raster* fonts.

This function must not be issued in an area bracket.

The initial default value of the character-break-extra attribute is 0, which gives normal spacing. This default value can be changed with GpiSetDefAttrs.

GpiSetCharBreakExtra — Set Character Break Extra

The attribute mode (see GpiSetAttrMode) determines whether the current value of the character-break-extra attribute is preserved.

Graphic Elements and Orders

Element Type: **OCODE_GSCBE**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Character Break Extra**

Element Type: **OCODE_GPSCBE**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Character Break Extra**

Example Code

This function specifies an extra increment to be used for spacing break characters in a string.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
```

```
HPS    hps;    /* Presentation space handle */
FIXED  fxBreak; /* Character-break-extra value. */
          /* world coordinates. */

fxBreak = 8L<<16; /* values are shifted to the */
                  /* to make them fixed. */

GpiSetCharBreakExtra(hps,
                    fxBreak);
```

GpiSetCharDirection — Set Character Direction

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetCharDirection (HPS hps, LONG IDirection)

This function determines the direction in which the characters in a string are drawn relative to the baseline.

Parameters

hps (HPS) — input

Presentation-space handle.

IDirection (LONG) — input

Character direction:

CHDIRN_DEFAULT The default; the same as CHDIRN_LEFTRIGHT (unless changed with GpiSetDefAttrs)

CHDIRN_LEFTRIGHT Character boxes are arranged parallel to, and in the same direction as, the baseline. This is the usual convention for Roman text.

CHDIRN_TOPBOTTOM Character boxes are arranged in columns directed 90 degrees *clockwise* from the baseline. This is the usual convention for Chinese characters. This option can be used for drawing Roman text vertically (a y-axis title on a graph, for example). The reference point within the character definition is at the center of the character, along the x-direction, in this case.

CHDIRN_RIGHTLEFT Character boxes are arranged parallel to, but in the reverse of, the baseline direction. This is the usual convention for Arabic text.

CHDIRN_BOTTOMTOP Character boxes are arranged in columns directed 90 degrees *counterclockwise* from the baseline. The reference point within the character definition is at the center of the character, along the x-direction, in this case.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_CHAR_DIRECTION_ATTR

An invalid character direction attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

GpiSetCharDirection — Set Character Direction

Remarks

This function must not be issued in an area bracket. The attribute mode determines whether the current value of the character direction attribute is preserved. This diagram shows how the origin of characters changes when the direction changes:

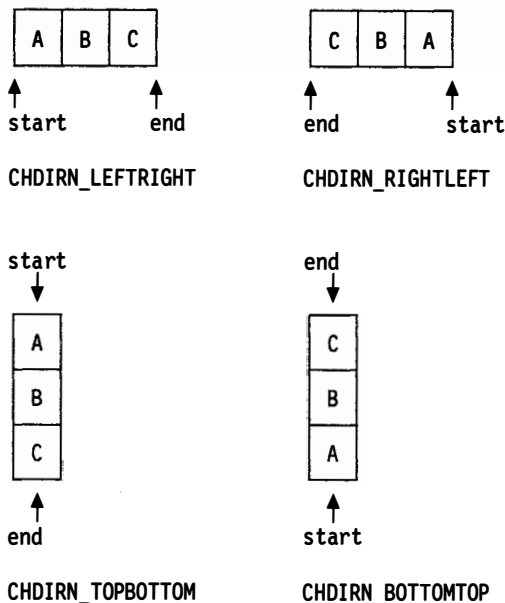


Figure 5-8. Character Drawing Directions and Character Box Origins

Related Functions

- GpiQueryCharDirection
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt

Graphic Elements and Orders

Element Type: **OCODE_GSCD**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Character Direction**

GpiSetCharDirection – Set Character Direction

Element Type: **OCODE_GPSCD**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Character Direction**

Example Code

This function determines the direction in which the characters in a string are drawn relative to the baseline. In this example, the direction is reset to the default, or left-to-right.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
```

```
HPS    hps;    /* Presentation space handle    */
```

```
GpiSetCharDirection(hps,
                     CHDIRN_DEFAULT);
```

GpiSetCharExtra — Set Character Extra

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetCharExtra (HPS hps, FIXED fxExtra)

This function specifies an extra increment to be used for spacing characters in a string.

Parameters

hps (HPS) — input
Presentation-space handle.

fxExtra (FIXED) — input
Character-extra value.

The value can be negative, 0, or positive:

- A negative value forces the characters closer together.
- A value of 0 results in normal spacing.
- A positive value forces the characters further apart.

The value is in world coordinates.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

Remarks

The character-extra attribute provides a spacing value that increases or decreases the spacing between characters in a string. It applies to all characters in a font, including the break character and is additional to the spacing generated for other reasons, for example:

- The spacing determined by the font, including proportional spacing and kerning, if applicable
- The vector of increment values, see `GpiCharStringPos`, `GpiCharStringPosAt`, `GpiQueryCharStringPos` and `GpiQueryCharStringPosAt`
- Break-extra spacing, see `GpiSetCharBreakExtra`.

Extra spacing applies to character strings either within or outside a path definition (see `GpiBeginPath`). The effect of the character-extra attribute applies whatever the value of the character-mode attribute (see `GpiSetCharMode`), and for both outline and *raster* fonts.

This function must not be issued in an area bracket.

The initial default value of the character-extra attribute is 0, which gives normal spacing. This default value can be changed with `GpiSetDefAttrs`.

The attribute mode (see `GpiSetAttrMode`) determines whether the current value of the character-extra attribute is preserved.

GpiSetCharExtra — Set Character Extra

Graphic Elements and Orders

Element Type: **OCODE_GSCE**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Character Extra**

Element Type: **OCODE_GPSCE**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Character Extra**

Example Code

This function specifies an extra increment to be used for spacing characters in a string.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
```

```
HPS    hps;    /* Presentation space handle    */
FIXED fxExtra; /* extra character.
```

```
fxExtra = MAKEFIXED(0,0) /* normal spacing. */
GpiSetCharExtra(hps, fxExtra);
```

GpiSetCharMode — Set Character Mode

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetCharMode (HPS hps, LONG IMode)

This function controls the character mode used when drawing a character string.

Parameters

hps (HPS) — input

Presentation-space handle.

IMode (LONG) — input

Character mode:

CM_DEFAULT The default; the same as CM_MODE1 (unless changed with GpiSetDefAttrs).

CM_MODE1 The font selected by means of GpiSetCharSet can be either a raster font or an outline font.

If it is a raster font, the position of characters after the first character is determined by the font metrics information, and by the character direction, character extra, and character break extra attributes. If it is an outline font, the behavior is as if the character mode is CM_MODE3.

CM_MODE2 The font selected by means of GpiSetCharSet can be either a raster font or an outline font.

If it is a raster font, the position of characters after the first character is determined by the font metrics information, and some character attributes, namely, GpiSetCharAngle, GpiSetCharBox, GpiSetCharDirection, GpiSetCharExtra, GpiSetCharBreakExtra, and GpiSetCharShear (the latter is relevant only for character directions of CHDIRN_TOPBOTTOM and CHDIRN_BOTTOMTOP). If it is an outline font, the behavior is as if the character mode is CM_MODE3.

CM_MODE3 All character attributes are used for positioning (together with the positioning information in the font), and for scaling, rotating, and shearing the characters.

The font selected by means of GpiSetCharSet must be an outline font; an error is raised if an attempt is made to draw a character string in CM_MODE3, and the selected font is a raster font.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_CHAR_MODE_ATTR

An invalid character mode attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

GpiSetCharMode – Set Character Mode

Remarks

The value of the *IMode* parameter controls whether the character attributes affect the positioning of characters, as follows:

Table 5-1. Use of Character Attributes in each Character Mode		
Character Mode	Font Type	Character Attributes (Angle, Shear, and Box)
Mode 1	Raster	Ignored
	Outline	Used
Mode 2	Raster	Attribute information is used to position characters; characters are not sheared, rotated, or scaled.
	Outline	Used
Mode 3	Raster	An error is raised when an attempt is made to draw a character string.
	Outline	Used

This function must not be issued in an area bracket.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the character-mode attribute is preserved.

Related Functions

- GpiQueryCharMode
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharSet
- GpiSetCharShear
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt

Graphic Elements and Orders

Element Type: **OCODE_GSCR**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Character Precision**

Element Type: **OCODE_GPSCR**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Character Precision**

GpiSetCharMode — Set Character Mode

Example Code

In this example the GpiSetCharMode call is used to set the character mode to raster or outline when drawing a string.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
```

```
HPS    hps;    /* Presentation space handle    */
```

```
GpiSetCharMode(hps,
                CM_MODE3); /* The font selected by    */
                          /* means of          */
                          /* GpiSetCharSet can be */
                          /* either a raster font or */
                          /* an outline font.      */
```

GpiSetCharSet – Set Character Set

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetCharSet (HPS hps, LONG lclid)

This function sets the current value of the character-set attribute.

Parameters

hps (HPS) – input

Presentation-space handle.

lclid (LONG) – input

Character-set local identifier:

LCID_DEFAULT Default (can be set explicitly with GpiSetDefAttrs).

1 – 254 Identifies a logical font.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_CHAR_SET_ATTR

An invalid character setid attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_HUGE_FONTS_NOT_SUPPORTED

An attempt was made using GpiSetCharSet, GpiSetPatternSet, GpiSetMarkerSet, or GpiSetAttrs to select a font that is larger than the maximum size (64Kb) supported by the target device driver.

Remarks

This function must not be issued in an area bracket.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the character-set attribute is preserved.

GpiSetCharSet — Set Character Set

Related Functions

- GpiCreateLogFont
- GpiQueryCharSet
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharShear
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt

Graphic Elements and Orders

Element Type: **OCODE_GSCS**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Character Set**

Element Type: **OCODE_GPSCS**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Character Set**

Example Code

This function sets the current value of the character-set attribute.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
```

```
HPS    hps;    /* Presentation space handle    */
LONG l1cid = 32L;
```

```
GpiSetCharSet(hps, l1cid);
```

GpiSetCharShear — Set Character Shear

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetCharShear (HPS hps, PPOINTL pptlAngle)

This function sets the character-shear attribute.

Parameters

hps (HPS) — input
Presentation-space handle.

pptlAngle (PPOINTL) — input
Character shear vector.

With reference to Figure 5-9 on page 5-446, the shear angle is defined in terms of the relative coordinates of the point *pptlAngle* (*x*, *y*).

If *x* is 0 and *y* is 1 (initial default), “upright” characters result. If *x* and *y* are both positive or both negative, the characters slope from bottom-left to top-right. If *x* and *y* are of opposite signs, the characters slope from top-left to bottom-right. No character inversion ever takes place as a result of a shear alone.

Usually, it is an error to specify 0 for *y*, because this implies an “infinite” shear. However, if both *x* and *y* are 0, the attribute is set to the default value. This can be changed from the initial default with GpiSetDefAttrs.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_CHAR_SHEAR_ATTR

An invalid character shear attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_COORDINATE

An invalid coordinate value was specified.

Remarks

The coordinates of the point *pptlAngle* (*x*, *y*), specify integer values that identify the end coordinates of a line originating at (0,0) (see Figure 5-9 on page 5-446). The vertical strokes in subsequent character strings are drawn parallel to the defined line. The top of the character box remains parallel to the character baseline (which may itself be rotated).

Whether this attribute is used when character strings are drawn depends on the type of font being used (raster or outline), and on the value of the character mode attribute (see GpiSetCharMode). If it is used, then with character directions of CHDIRN_TOPBOTTOM and CHDIRN_BOTTOMTOP (see GpiSetCharDirection) the whole string is tilted by the shear angle, in addition to the individual characters being sheared if the current font is an outline font.

This function must not be issued in an area bracket.

GpiSetCharShear — Set Character Shear

The attribute mode (see GpiSetAttrMode) determines whether the current value of the character shear attribute is preserved.

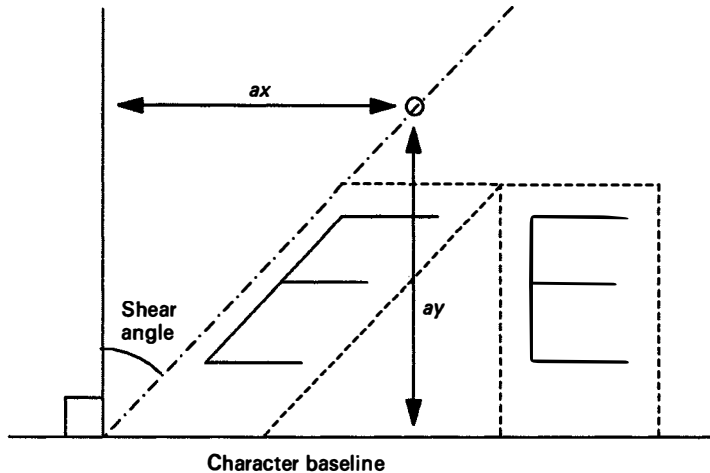


Figure 5-9. Character Shear

Related Functions

- GpiQueryCharShear
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt

Graphic Elements and Orders

Element Type: `OCODE_GSCH`

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to `AM_NOPRESERVE`.

Order: `Set Character Shear`

Element Type: `OCODE_GPSCH`

This element type is generated if the attribute mode is set to `AM_PRESERVE`.

Order: `Push and Set Character Shear`

GpiSetCharShear – Set Character Shear

Example Code

This function sets the character-shear attribute.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS    hps;    /* Presentation space handle */
POINTL ptlAngle = {50L, 70L}; /* character shear vector. */

GpiSetCharShear(hps,
                &ptlAngle); /* the shear */
                           /* angle is defined in terms */
                           /* of the relative */
                           /* coordinates of the point */
                           /* pptlAngle. This can be */
                           /* changed from the initial */
                           /* default with */
                           /* GpiSetDefAttrs. */
```

GpiSetClipPath — Set Clip Path

```
#define INCL_GPIPATHS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetClipPath (HPS hps, LONG IPath, LONG IOptions)

This function selects a path as the current clip path.

Parameters

hps (HPS) — input
Presentation-space handle.

IPath (LONG) — input
Path control flag.

0 The current clip path stops being the current clip path; the current clip path is to be reset to an infinite one (that is, no clipping).

1 The path that has been defined is to be intersected with the current clip path.

IOptions (LONG) — input
Options.

This contains fields of option bits. For each field, one value should be selected (unless the default is suitable). These values can then be ORed together to generate the parameter.

How to construct the path interior (see also GpiBeginArea):

SCP_ALTERNATE Construct interior in alternate mode.

SCP_WINDING Construct interior in winding mode. This value must be selected if the path has been modified using GpiModifyPath.

The default is SCP_ALTERNATE.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PATH_ID An invalid path identifier parameter was specified.

PMERR_INV_CLIP_PATH_OPTIONS An invalid options parameter was specified with GpiSetClipPath.

PMERR_PATH_UNKNOWN An attempt was made to perform a path function on a path that did not exist.

GpiSetClipPath — Set Clip Path

Remarks

The clip path (bound in device coordinates when the path is defined) is used for all subsequent drawing.

Any open figures within the path are closed automatically.

The boundaries of the area defined by the path are considered to be part of the interior, so that a point on the boundary is not clipped.

The clip path is reset to no clipping (no path selected) at the start of a root segment (subject to the fast chaining segment attribute), or when a GpiResetPS function is issued.

After a path is selected as the clip path, it cannot be reused for any other purpose. When it is superseded as the clip path, it is discarded.

Related Functions

- GpiBeginPath
- GpiEndPath
- GpiFillPath
- GpiModifyPath
- GpiOutlinePath
- GpiPathToRegion
- GpiStrokePath
- GpiExcludeClipRectangle
- GpiIntersectClipRectangle
- GpiOffsetClipRegion
- GpiQueryClipBox
- GpiQueryClipRegion
- GpiSetClipPath
- GpiSetClipRegion

Graphic Elements and Orders

Element Type: OCODE_GSCPTH

Order: Select Clip Path

GpiSetClipPath — Set Clip Path

Example Code

This function selects a path as the current clip path.

```
#define INCL_GPIPATHS
#include <OS2.H>
```

```
HPS    hps;    /* Presentation space handle */
```

```
GpiSetClipPath(hps,
               0L, /* The current clip path */
                  /* stops being the */
                  /* current clip path; the */
                  /* current clip path is to */
                  /* be reset to an infinite */
                  /* one (that is, no */
                  /* clipping) */
               SCP_ALTERNATE);
                  /* Construct interior in */
                  /* alternate mode. */
```

GpiSetClipRegion – Set Clip Region

```
#define INCL_GPIREGIONS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiSetClipRegion (HPS hps, HRGN hrgn, PHRGN phrgnOld)

This function defines the region to be used for clipping, when any drawing takes place through the specified presentation space.

Parameters

hps (HPS) – input

Presentation-space handle.

The presentation space must be currently associated with a device context of the correct device class (defined when the region is first created).

hrgn (HRGN) – input

Region handle.

If *hrgn* is null, the clipping region is set to no clipping (its initial state).

phrgnOld (PHRGN) – output

Old region handle (if any):

HRGN_ERROR Error

NULLHANDLE Null handle (no region selected)

Otherwise Old region handle.

Returns

Complexity of clipping and error indicators:

The clipping complexity information includes the combined effects of:

- Clip path
- Viewing limits
- Graphics field
- Clip region
- Visible region (windowing considerations).

RGN_NULL Null region

RGN_RECT Rectangular region

RGN_COMPLEX Complex region

RGN_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HRGN

An invalid region handle was specified.

PMERR_HRGN_BUSY

An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

GpiSetClipRegion —

Set Clip Region

Remarks

While a region is in use as a clip region, the calls `GpiOffsetClipRegion`, `GpiExcludeClipRectangle` and `GpiIntersectClipRectangle` cause it to be changed. These changes persist after the region has been deselected. The clip region cannot, however, be used for any other region operations, nor can it be selected into any other presentation space as a clipping region, until it is deselected.

The coordinates of the region are taken to be device coordinates within the device context.

The previous clip region, if any, is converted to a region, and a handle to it is returned. This can be used in a subsequent `GpiSetClipRegion` to reinstate the same clipping as before. If no clip region exists, a null handle is returned. It is the responsibility of the application to free the previous clip region (if any) with `GpiDestroyRegion`, even if this region was not originally created explicitly by the application.

Note: This function must not be used when creating SAA-conforming metafiles; see “Metafile Restrictions” on page G-1.

Related Functions

- `GpiCreateRegion`
- `GpiExcludeClipRectangle`
- `GpiIntersectClipRectangle`
- `GpiOffsetClipRegion`
- `GpiPtVisible`
- `GpiQueryClipBox`
- `GpiQueryClipRegion`
- `GpiRectVisible`
- `WinExcludeUpdateRegion`

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

```
BOOL GpiSetColor (HPS hps, LONG IColor)
```

This function sets the current value of the color attribute for each of the individual primitive types.

Parameters

hps (HPS) – input
Presentation-space handle.

IColor (LONG) – input
Color:

CLR_FALSE	All color planes are 0s.
CLR_TRUE	All color planes are 1s.
CLR_DEFAULT	Set to default value. This is a device-dependent color which, for the default color table, provides a contrasting color to CLR_BACKGROUND. For a display, it is the default window text color (SYSCLR_WINDOWTEXT: see WinSetSysColors). For a printer, it is a color that contrasts with the paper color. The default can be changed by setting new system colors from the control panel for the display, or by selecting a paper color for a printer, if allowed by the device driver. It can also be set explicitly, using GpiSetDefAttrs.
CLR_WHITE	White (default color table, or index=RGB loaded color table). For a loaded, realized, color table it is the nearest available color to white.
CLR_BLACK	Black (default color table, or index=RGB loaded color table). For a loaded, realized, color table, it is the nearest available color to black.
CLR_BACKGROUND	Reset color, used by GpiErase. This is the natural background color for the device. For a display, it is the default window color (SYSCLR_WINDOW: see WinSetSysColors) for the default color table. For a printer, it is the paper color. For a loaded color table, it is color index 0. For an RGB color table, it is color 000000 (black).
CLR_BLUE	Blue (default color table).
CLR_RED	Red (default color table).
CLR_PINK	Pink (default color table).
CLR_GREEN	Green (default color table).
CLR_CYAN	Cyan (default color table).
CLR_YELLOW	Yellow (default color table).
CLR_NEUTRAL	Neutral (default color table). A device-dependent color, that for the default color table provides a contrasting color to CLR_BACKGROUND. For a display, it is the default window text color (SYSCLR_WINDOWTEXT: see WinSetSysColors). For a printer, it is a color that contrasts with the paper color. For a loaded color table, it is color index 7; in RGB mode it is interpreted as color 000007.
CLR_DARKGRAY	Dark gray (default color table).
CLR_DARKBLUE	Dark blue (default color table).
CLR_DARKRED	Dark red (default color table).

GpiSetColor — Set Color

CLR_DARKPINK	Dark pink (default color table).
CLR_DARKGREEN	Dark green (default color table).
CLR_DARKCYAN	Dark cyan (default color table).
CLR_BROWN	Brown (default color table).
CLR_PALEGRAY	Pale gray (default color table).

For a loadable color table, values 0 through n correspond to the color index (or RGB) values.

Returns

Success indicator:

TRUE	Successful completion
FALSE	Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_COLOR_ATTR	An invalid color attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

Remarks

The current values for each primitive type are updated. The attribute mode (see GpiSetAttrMode) determines whether the current values of the individual color attributes are preserved. If so, they are restored by a single GpiPop function.

An attempt to set a negative color value, other than one for which a constant is defined, causes the error PMERR_INV_COLOR_ATTR to be logged. Other color values are allowed, although an error is generated when the color value is needed for drawing if it is not valid for the color table in use at that time (see GpiCreateLogColorTable).

For details of how colors are handled on monochrome devices, see GpiCreateLogColorTable.

Related Functions

- GpiQueryColor
- GpiSetBackColor
- GpiSetBackMix
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetMix
- WinSetSysColors

Graphic Elements and Orders

Element Type: **OCODE_GSICOL**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Indexed Color**

Element Type: **OCODE_GPSICOL**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Indexed Color**

Example Code

This example draws a blue line.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
```

```
HPS    hps;    /* Presentation space handle    */
POINTL pt11, pt12;

    GpiSetColor(hps, CLR_BLUE);
    GpiMove( hps, &pt11 );          /* Move to start point    */
    GpiLine( hps, &pt12 );          /* Draw new line          */
```

GpiSetCp — Set Code Page

```
#define INCL_GPILCIDS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetCp (HPS hps, ULONG ulCodePage)

This function sets the default graphics code page.

Parameters

hps (HPS) — input
Presentation-space handle.

ulCodePage (ULONG) — input
Code-page identifier.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_CODEPAGE	An invalid code-page parameter was specified with GpiSetCp.

Remarks

The default graphics code page is used for the default font (unless it is overridden by GpiCreateLogFont). It is also used for other fonts for which the *usCodePage* field in the FATTRS structure for GpiCreateLogFont is specified as 0. This includes existing fonts that are defined in this way.

Any code page that is defined in the CONFIG.SYS file, or is a supported EBCDIC code page, can be selected.

The list of available code pages is returned by WinQueryCpList.

When a GPI presentation space is first created, the code page in force is that defined by the process code page.

If this function occurs within a path definition when the drawing mode (see GpiSetDrawingMode) is **retain** or **draw-and-retain**, its effect is not stored with the definition.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see "Metafile Restrictions" on page G-1.

Related Functions

The DOS calls `DosGetCp`, `DosSetCp`, and `DosSetProcCp` are related to `GpiSetCP`, but they are not a part of the Presentation Manager, for more information on the mentioned DOS calls refer to the Control Program Reference.

- `GpiQueryCurrentPosition`
- `GpiQueryCurrentPosition`
- `GpiCreateLogFont`
- `WinCpTranslateChar`
- `WinCpTranslateString`
- `WinQueryCp`
- `WinQueryCpList`
- `WinSetCp`

Example Code

This example sets the code page to 850.

```
#define INCL_GPILCIDS  
#include <OS2.H>
```

```
HPS    hps;    /* Presentation space handle    */  
ULONG  ulCodePage = 850;
```

```
GpiSetCp(hps, ulCodePage);
```

GpiSetCurrentPosition — Set Current Position

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetCurrentPosition (HPS hps, PPOINTL pptlPoint)

This function sets the current position to the specified point.

Parameters

hps (HPS) — input
Presentation-space handle.

pptlPoint (PPOINTL) — input
New value of current position.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.

Remarks

This function also has the effect of resetting the position within a line-type sequence, and, if within an area, of starting a new closed figure and causing any previous one to be closed if necessary.

This function is equivalent to the GpiMove function except that, if the current attribute mode is AM_PRESERVE (see GpiSetAttrMode), the current position is saved before being set to the new value, so that it can be restored using the GpiPop function.

Related Functions

- GpiMove
- GpiQueryCurrentPosition
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs

Graphic Elements and Orders

Element Type: **OCODE_GSCP**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Current Position**

Element Type: **OCODE_GPSCP**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Current Position**

GpiSetCurrentPosition – Set Current Position

Example Code

The position of the top-left corner of the window rectangle is recorded and selected as the current position before the image is drawn.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
HPS      hps;      /* Presentation space handle      */
HWND hwndClient; /* client window handle. */
RECTL rcl;
POINTL vptlSave;

WinQueryWindowRect(hwndClient, &rcl);
vptlSave.x = rcl.xLeft;
vptlSave.y = rcl.yTop;
GpiSetCurrentPosition(hps, &vptlSave);
```


GpiSetDefArcParams – Set Default Arc Parameters

```
#define INCL_GPIDEFAULTS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetDefArcParams (HPS hps, PARCPARAMS parcpArcParams)

This function specifies the default values of the arc parameters (see GpiSetArcParams).

Parameters

hps (HPS) – input

Presentation-space handle.

parcpArcParams (PARCPARAMS) – input

Default arc parameters.

This structure has four elements p, q, r, and s.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE

An invalid coordinate value was specified.

Remarks

The arc parameters are reset to their default values at the following times:

- When the presentation space is associated with a device context (see GpiAssociate).
- When GpiResetPS is issued.
- When drawing of a chained segment begins or ends (see GpiOpenSegment and GpiCloseSegment for more details).

The initial default values of the arc parameters, when the presentation space is first created, are :

p = 1 r = 0
s = 0 q = 1

The default values can be changed by GpiSetDefArcParams. Changing the default values has an immediate effect on the current arc parameters, if these are currently set to the default value.

See “GpiSetArcParams – Set Arc Parameters” on page 5-398 for a description of the arc parameters.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see “Metafile Restrictions” on page G-1.

GpiSetDefArcParams — Set Default Arc Parameters

Related Functions

- GpiFullArc
- GpiPartialArc
- GpiPointArc
- GpiQueryDefArcParams
- GpiSetArcParams
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs

Example Code

This function specifies the default values of the arc parameters (see GpiSetArcParams).

```
#define INCL_GPIDEFAULTS
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS    hps;    /* Presentation space handle    */

ARCPARAMS ArcParams = {10L, /* p */
                        20L, /* q */
                        10L, /* r */
                        30L}; /* l */
                        /* This structure has four */
                        /* elements p, q, r, and s. */

GpiSetDefArcParams(hps, &ArcParams);
```

GpiSetDefAttrs — Set Default Attributes

```
#define INCL_GPIDEFAULTS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetDefAttrs (HPS hps, LONG IPrimType, ULONG flAttrMask, PBUNDLE ppbunAttrs)

This function sets the default values of attributes for the specified primitive type.

Parameters

hps (HPS) — input
Presentation-space handle.

IPrimType (LONG) — input
Primitive type.

The primitive type for which default attributes are to be set:

PRIM_LINE Line and arc primitives

PRIM_CHAR Character primitives

PRIM_MARKER Marker primitives

PRIM_AREA Area primitives

PRIM_IMAGE Image primitives.

flAttrMask (ULONG) — input
Attributes mask.

Each flag that is set indicates that the *ppbunAttrs* buffer contains data for the corresponding attribute. If all the flags in *flAttrMask* are 0, the *ppbunAttrs* buffer address is not used.

Line attributes:

LBB_COLOR Line color

LBB_MIX_MODE Line mix

LBB_WIDTH Line width

LBB_GEOM_WIDTH Geometric line width

LBB_TYPE Line type

LBB_END Line end

LBB_JOIN Line join.

Character attributes:

CBB_COLOR Character color

CBB_BACK_COLOR Character background color

CBB_MIX_MODE Character mix

CBB_BACK_MIX_MODE Character background mix

CBB_SET Character set

CBB_MODE Character mode

CBB_BOX Character box

CBB_ANGLE Character angle

CBB_SHEAR Character shear

CBB_DIRECTION Character direction

GpiSetDefAttrs – Set Default Attributes

CBB_EXTRA	Character extra
CBB_BREAK_EXTRA	Character extra
Marker attributes:	
MBB_COLOR	Marker color
MBB_BACK_COLOR	Marker background color
MBB_MIX_MODE	Marker mix
MBB_BACK_MIX_MODE	Marker background mix
MBB_SET	Marker set
MBB_SYMBOL	Marker symbol
MBB_BOX	Marker box.
Pattern attributes (areas):	
ABB_COLOR	Area color
ABB_BACK_COLOR	Area background color
ABB_MIX_MODE	Area mix
ABB_BACK_MIX_MODE	Area background mix
ABB_SET	Pattern set
ABB_SYMBOL	Pattern symbol
ABB_REF_POINT	Pattern reference point.
Image attributes:	
IBB_COLOR	Image color
IBB_BACK_COLOR	Image background color
IBB_MIX_MODE	Image mix
IBB_BACK_MIX_MODE	Image background mix.

ppbunAttrs (PBUNDLE) – input
Default attribute values.

This is a structure containing default attribute values for each attribute for which the *flAttrMask* flag is set, at the correct offset as specified below for the particular primitive type.

Line attributes: *ppbunAttrs* consists of a LINEBUNDLE structure.

Character attributes: *ppbunAttrs* consists of a CHARBUNDLE structure.

Marker attributes: *ppbunAttrs* consists of a MARKERBUNDLE structure.

Pattern attributes (areas): *ppbunAttrs* consists of an AREABUNDLE structure.

Image attributes: *ppbunAttrs* consists of an IMAGEBUNDLE structure.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

GpiSetDefAttrs – Set Default Attributes

PMERR_INV_PRIMITIVE_TYPE	An invalid primitive type parameter was specified with GpiSetAttrs or GpiQueryAttrs.
PMERR_UNSUPPORTED_ATTR	An unsupported attribute was specified in the attrmask with GpiSetAttrs or GpiQueryAttrs.
PMERR_INV_COLOR_ATTR	An invalid color attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_BACKGROUND_COL_ATTR	An invalid background color attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_MIX_ATTR	An invalid mix attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_LINE_WIDTH_ATTR	An invalid line width attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_GEOM_LINE_WIDTH_ATTR	An invalid geometric line width attribute value was specified.
PMERR_INV_LINE_TYPE_ATTR	An invalid line type attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_LINE_END_ATTR	An invalid line end attribute value was specified.
PMERR_INV_LINE_JOIN_ATTR	An invalid line join attribute value was specified.
PMERR_INV_CHAR_SET_ATTR	An invalid character setid attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_CHAR_MODE_ATTR	An invalid character mode attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_CHAR_DIRECTION_ATTR	An invalid character direction attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_CHAR_SHEAR_ATTR	An invalid character shear attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_CHAR_ANGLE_ATTR	The default character angle attribute value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_MARKER_SET_ATTR	An invalid marker set attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_MARKER_SYMBOL_ATTR	An invalid marker symbol attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_PATTERN_SET_ATTR	An invalid pattern set attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_PATTERN_ATTR	An invalid pattern symbol attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

GpiSetDefAttrs — Set Default Attributes

PMERR_INV_COORDINATE	An invalid coordinate value was specified.
PMERR_UNSUPPORTED_ATTR_VALUE	An attribute value was specified with GpiSetAttrs that is not supported.
PMERR_INV_PATTERN_SET_FONT	An attempt was made to use an unsuitable font as a pattern set.
PMERR_HUGE_FONTS_NOT_SUPPORTED	An attempt was made using GpiSetCharSet, GpiSetPatternSet, GpiSetMarkerSet, or GpiSetAttrs to select a font that is larger than the maximum size (64Kb) supported by the target device driver.

Remarks

Attributes are reset to their default values at the following times:

- When the presentation space is associated with a device context (see GpiAssociate).
- When GpiResetPS is issued.
- When drawing of a chained segment begins or ends (see GpiOpenSegment and GpiCloseSegment for more details).
- When an attribute-setting function (for example, GpiSetAttrs) that sets an attribute to its default value is issued, or interpreted in a retained segment during a drawing operation.

Each attribute has an initial default value, established when the presentation space is first created. The value of this is given under the appropriate GpiSet... call. The default value can be changed by GpiSetDefAttrs. Changing the default value takes effect immediately for the current value, if this is set to default at the time.

Each attribute of the primitive type in question is represented by one flag in the *flAttrMask* parameter. Any attribute for which the appropriate flag is set has its default value updated to the value specified in the *ppbunAttrs* structure. If the attribute is currently set to take the default value, it is immediately assigned the new default value. The default value of any attribute for which the appropriate flag in *flAttrMask* is not set is unchanged.

The data in the *ppbunAttrs* buffer consists of a structure of attribute data. The layout of the structure is fixed for each primitive type. Only data for attributes for which the flag is set in *flAttrMask* is inspected; any other data is ignored.

Note: The buffer need be no longer than is necessary to contain the data for the highest offset attribute referenced.

If an attempt is made to set an invalid default value by this function, none of the specified default attribute values is changed. Note, however, that some invalid default attribute values (for example, certain color and mix values) may not be detected until the attribute is set to default and used, at which point the implementation optionally defaults them, or causes an error to be logged.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see "Metafile Restrictions" on page G-1. Also, in a metafile, the default line width (see GpiSetLineWidth) is always rounded to an integer value, as is the default character box (see GpiSetCharBox) for GPIF_SHORT format presentation spaces (see GpiCreatePS).

GpiSetDefAttrs — Set Default Attributes

Example Code

This function sets the default color of line and arc primitives to blue.

```
#define INCL_GPIDEFAULTS
#define INCL_GPIPRIMITIVES /* for parameter definitions */
#include <OS2.H>
HPS      hps; /* Presentation space handle */
LINEBUNDLE bunAttrs; /* Information for color */

bunAttrs.lColor = CLR_BLUE;

GpiSetDefAttrs(hps,
               PRIM_LINE, /* line and arc primitives. */
               LBB_COLOR, /* color information, which is */
                       /* contained in bunAttrs */
               &bunAttrs);
```

GpiSetDefaultViewMatrix – Set Default View Matrix

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
```

```
BOOL GpiSetDefaultViewMatrix (HPS hps, LONG ICount, PMATRIXLF pmatlfarray,  
                             LONG IOptions)
```

This function sets the default viewing transform that is to apply to the whole picture.

Parameters

hps (HPS) – input
Presentation-space handle.

ICount (LONG) – input
Number of elements.

The number of elements supplied in *pmatlfarray*, that are to be examined, starting from the beginning of the structure. If *ICount* is less than 9, remaining elements default to the corresponding elements of the identity matrix. Specifying *ICount* = 0 means that the identity matrix is used.

pmatlfarray (PMATRIXLF) – input
Transformation matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type FIXED, and have an assumed binary point between the second and third bytes. Thus a value of 1.0 is represented by 65 536. Other elements are normal signed integers. If the presentation-space coordinate format is GPIF_SHORT (see GpiCreatePS), these elements must be within the range –1 through +1.

The third, sixth, and ninth elements, when specified, must be 0, 0, and 1, respectively.

IOptions (LONG) – input
Transform options.

Specifies how the transform defined by the *pmatlfarray* parameter should be used to modify the existing default viewing transform.

Possible values are:

- | | |
|--------------------------|--|
| TRANSFORM_REPLACE | The previous default viewing transform is discarded and replaced by the specified transform. |
| TRANSFORM_ADD | The specified transform is combined with the existing default viewing transform, in the order (1) existing transform, (2) new transform. This option is most useful for incremental updates to transforms. |
| TRANSFORM_PREEMPT | The specified transform is combined with the existing default viewing transform, in the order (1) new transform, (2) existing transform. |

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

GpiSetDefaultViewMatrix —

Set Default View Matrix

PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_INV_TRANSFORM_TYPE	An invalid options parameter was specified with a transform matrix function.
PMERR_INV_MATRIX_ELEMENT	An invalid transformation matrix element was specified.

Remarks

The transform matrix specified is used to update any previous default viewing transform, depending upon the value of *IOptions*.

The transform is specified as a one-dimensional array of *ICount* elements, being the first n elements of a 3-row by 3-column matrix ordered by rows. The order of the elements is:

Matrix	Array
$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix}$	(a,b,0,c,d,0,e,f,1)

The transform acts on the coordinates of the primitives in a segment, so that a point with coordinates (x,y) is transformed to the point:

(a*x + c*y + e, b*x + d*y + f)

The initial value of the default viewing transform is the identity matrix, as shown below:

Matrix	Array
$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	(1,0,0,0,1,0,0,0,1)

If scaling values greater than unity are given (which only applies if the presentation space coordinate format, as set by the GpiCreatePS function, is GPIF_LONG), it is possible for the combined effect of this and any other relevant transforms to exceed fixed-point implementation limits. This causes an error.

This function must not be issued in a path or area bracket.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see "Metafile Restrictions" on page G-1.

Related Functions

- GpiQueryDefaultViewMatrix
- GpiSetViewingTransformMatrix

GpiSetDefaultViewMatrix — Set Default View Matrix

Example Code

This example uses the GpiSetDefaultViewMatrix function to replace the existing default viewing transformation. The new transformation translates drawing to the right by 100 units.

```
#define INCL_GPITRANSFORMS    /* Transform functions    */
#include <os2.h>

BOOL fSuccess;               /* success indicator    */
HPS  hps;                    /* Presentation-space handle */
/* transform matrix */
MATRIXLF matlf = {MAKEFIXED(1,0), 0, 0, 0, MAKEFIXED(1,0), 0, 100};

fSuccess = GpiSetDefaultViewMatrix(hps, 7L, &matlf,
                                   TRANSFORM_REPLACE);
```

GpiSetDefTag — Set Default Tag

```
#define INCL_GPIDEFAULTS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetDefTag (HPS hps, LONG ITag)

This function specifies the default value of the primitive tag (see GpiSetTag).

Parameters

hps (HPS) — input
Presentation-space handle.

ITag (LONG) — input
Default tag identifier.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.

Remarks

The primitive tag is reset to its default value at the following times:

- When the presentation space is associated with a device context (see GpiAssociate).
- When GpiResetPS is issued.
- When drawing of a chained segment begins or ends (see GpiOpenSegment and GpiCloseSegment for more details).

The initial default value of the primitive tag, when the presentation space is first created, is 0. The default value can be changed by GpiSetDefTag. Changing the default value has an immediate effect on the current primitive tag, if this is currently set to the default value.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see “Metafile Restrictions” on page G-1.

Related Functions

- GpiQueryDefAttrs
- GpiQueryTag
- GpiSetTag

GpiSetDefTag – Set Default Tag

Example Code

This function specifies the default value of the primitive tag (see GpiSetTag).

```
#define INCL_GPIPRIMITIVES  
#include <OS2.H>
```

```
HPS    hps;    /* Presentation space handle    */
```

```
GpiSetDefTag(hps, 1L);
```

GpiSetDefViewingLimits – Set Default Viewing Limits

```
#define INCL_GPIDEFAULTS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetDefViewingLimits (HPS hps, PRECTL prclLimits)

This function specifies the default value of the viewing limits (see GpiSetViewingLimits).

Parameters

hps (HPS) – input
Presentation-space handle.

prclLimits (PRECTL) – input
Default viewing limits.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.

Remarks

The viewing limits are reset to their default value at the following times:

- When the presentation space is associated with a device context (see GpiAssociate).
- When GpiResetPS is issued.
- When drawing of a chained segment begins or ends (see GpiOpenSegment and GpiCloseSegment for more details).

The initial default value of the viewing limits, when the presentation space is first created, is no clipping. The default value can be changed by GpiSetDefViewingLimits. Changing the default values has an immediate effect on the current viewing limits, if these are currently set to the default value.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see “Metafile Restrictions” on page G-1.

Related Functions

- GpiQueryDefViewingLimits
- GpiQueryGraphicsField
- GpiQueryViewingLimits
- GpiSetGraphicsField
- GpiSetViewingLimits

GpiSetDefViewingLimits – Set Default Viewing Limits

Example Code

In this example the default model space clipping region width is set to 100.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;          /* Presentation-space */
                  /* handle.          */
RECTL rcLimits; /* viewing limits.    */

rcLimits.xRight = 100;
rcLimits.xLeft = 100;

GpiSetDefViewingLimits(hps,
                       &rcLimits);
```

GpiSetDrawControl —

Set Draw Control

```
#define INCL_GPICONTROL /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetDrawControl (HPS hps, LONG IControl, LONG IValue)

This function sets options for subsequent drawing operations.

Parameters

hps (HPS) — input
Presentation-space handle.

IControl (LONG) — input
Drawing control.

Note: Controls identified by an asterisk (*) are the only ones relevant to a micro-presentation space. Any other control settings are ignored for a micro-presentation space.

DCTL_ERASE Erase before draw. Perform an implicit GpiErase operation before GpiDrawChain, GpiDrawFrom, or GpiDrawSegment. The output display area of the Device Context associated with the specified presentation space is cleared before drawing.

DCTL_DISPLAY (*) Enable drawing to occur on the output medium. If this control is set to off, then except for GpiErase, no output operations appear on the output medium. This includes raster operations, such as drawing primitives, and GpiDraw... operations.

DCTL_BOUNDARY (*) Accumulate boundary data. During any output operations except GpiErase, accumulate the bounding rectangle of the drawing.

DCTL_DYNAMIC Draw dynamic segments. Perform an implicit GpiRemoveDynamics before GpiDrawChain, GpiDrawFrom, or GpiDrawSegment, and an implicit GpiDrawDynamics afterwards.

Note that, to either remove or draw dynamic segments, the system forces the foreground mix to FM_XOR, and the background mix to BM_LEAVEALONE. If the first nondynamic segment being drawn (immediately after the dynamic segments have been removed) has the ATTR_FASTCHAIN attribute (see GpiSetInitialSegmentAttrs), it may be necessary for it to set the mix modes itself before drawing. Similar considerations might apply for any subsequent drawing after the dynamic segments have been replaced.

DCTL_CORRELATE (*) If this control is set, any GpiPutData, GpiElement, GpiPlayMetaFile, or individual drawing primitives which are passed across the API outside a segment bracket cause a correlation operation to be performed, and a return code to be set if a hit occurs. (Correlation inside segments, both retained and nonretained, is controlled by the segment attribute ATTR_DETECTABLE).

This control has an effect only in **draw** or **draw-and-retain** modes (see GpiSetDrawingMode).

IValue (LONG) — input
Required value of the drawing control:

DCTL_OFF Set control off

DCTL_ON Set control on.

GpiSetDrawControl — Set Draw Control

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_DRAW_CONTROL	An invalid control parameter was specified with GpiSetDrawControl or GpiQueryDrawControl.
PMERR_INV_DRAW_VALUE	An invalid value parameter was specified with GpiSetDrawControl.
PMERR_INV_IN_SEG	An attempt was made to issue a function invalid inside a segment bracket.
PMERR_INV_IN_AREA	An attempt was made to issue a function invalid inside an area bracket. This can be detected while the actual drawing mode is draw or draw-and-retain or during segment drawing or correlation functions.
PMERR_INV_IN_PATH	An attempt was made to issue a function invalid inside a path bracket.
PMERR_INV_IN_ELEMENT	An attempt was made to issue a function invalid inside an element bracket.
PMERR_INV_MICROPS_DRAW_CONTROL	A draw control parameter was specified with GpiSetDrawControl that is invalid in a micro presentation space.

Remarks

The default value is DCTL_OFF for all controls except DCTL_DISPLAY (*). Its default value is DCTL_ON.

This function must not be issued in any of these cases:

- Inside an open segment
- Outside an open segment, but inside one of:
 - Area bracket
 - Element bracket
 - Path bracket.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see "Metafile Restrictions" on page G-1.

GpiSetDrawControl — Set Draw Control

Related Functions

- GpiDrawChain
- GpiDrawDynamics
- GpiDrawFrom
- GpiDrawSegment
- GpiErase
- GpiQueryDrawControl
- GpiQueryDrawingMode
- GpiQueryStopDraw
- GpiRemoveDynamics
- GpiSetDrawingMode
- GpiSetStopDraw

Example Code

The “display” drawing control is switched off, and the “accumulate-boundary-data” control is switched on.

```
#define INCL_GPICONTROL
#include <OS2.H>

HPS hps;          /* Presentation-space */
                  /* handle.          */

GpiResetBoundaryData(hps);
GpiSetDrawControl(hps, DCTL_DISPLAY, DCTL_OFF);
```

GpiSetDrawingMode – Set Drawing Mode

```
#define INCL_GPICONTROL /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetDrawingMode (HPS hps, LONG IMode)

This function sets the drawing mode to control the handling of subsequent individual drawing primitive and attribute calls.

Parameters

hps (HPS) – input

Presentation-space handle.

IMode (LONG) – input

Mode to be used for subsequent drawing calls:

DM_DRAW Draw, unless in an unchained segment

DM_RETAIN Retain, if within a segment

DM_DRAWANDRETAIN Draw-and-retain, combination of above.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION

An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_IN_AREA

An attempt was made to issue a function invalid inside an area bracket. This can be detected while the actual drawing mode is **draw** or **draw-and-retain** or during segment drawing or correlation functions.

PMERR_INV_IN_PATH

An attempt was made to issue a function invalid inside a path bracket.

PMERR_INV_IN_ELEMENT

An attempt was made to issue a function invalid inside an element bracket.

PMERR_INV_IN_SEG

An attempt was made to issue a function invalid inside a segment bracket.

PMERR_INV_DRAWING_MODE

An invalid mode parameter was specified with GpiSetDrawControl not **draw-and-retain** or **draw**.

GpiSetDrawingMode — Set Drawing Mode

Remarks

The drawing mode affects the handling of subsequent individual drawing primitive and attribute calls, and the GpiPutData, GpiElement, and GpiPlayMetaFile functions.

Primitives and attributes can be drawn immediately, retained, or both, in the current segment.

Note: Any primitive and attribute setting calls that occur **outside** a segment (that is, outside a GpiOpenSegment — GpiCloseSegment bracket) are always treated as nonretained. Conversely, any segments that are not chained are always retained. This table summarizes how the actual drawing mode is arrived at:

GpiSetDrawingMode Parameter	Context		
	Chained Segment	Unchained Segment	Outside Segment
DM_DRAWANDRETAIN	draw-and-retain	retain	draw
DM_RETAIN	retain	retain	draw
DM_DRAW	draw	retain	draw

The actual drawing mode (referred to when describing other Gpi calls) therefore depends upon the mode as set by GpiSetDrawingMode, together with the context, as in the table. It is this actual drawing mode that determines whether a drawing call is retained (**retain** or **draw-and-retain**), and whether it is drawn immediately (**draw** or **draw-and-retain**).

It is an error to try to set the drawing mode within a segment bracket, and also outside a segment bracket, if in one of the following:

- Area bracket
- Element bracket
- Path bracket.

The default drawing mode is DM_DRAW.

Related Functions

- GpiDrawChain
- GpiDrawDynamics
- GpiDrawFrom
- GpiDrawSegment
- GpiErase
- GpiGetData
- GpiOpenSegment
- GpiPutData
- GpiQueryDrawControl
- GpiQueryDrawingMode
- GpiQueryStopDraw
- GpiRemoveDynamics
- GpiSetStopDraw
- GpiOpenSegment

GpiSetDrawingMode – Set Drawing Mode

Example Code

This example calls GpiSetDrawingMode to set the drawing mode to DRAW.

```
#define INCL_GPICONTROL      /* GPI control Functions      */
#include <os2.h>

BOOL fSuccess;              /* success indicator    */
HPS  hps;                   /* Presentation-space handle */

fSuccess = GpiSetDrawingMode(hps, DM_DRAW);
```

GpiSetEditMode — Set Edit Mode

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetEditMode (HPS hps, LONG IMode)

This function sets the current editing mode.

Parameters

hps (HPS) — input
Presentation-space handle.

IMode (LONG) — input
Edit mode:

SEGEM_INSERT Insert mode

SEGEM_REPLACE Replace mode.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_EDIT_MODE An invalid mode parameter was specified with GpiSetEditMode.

PMERR_INV_IN_ELEMENT An attempt was made to issue a function invalid inside an element bracket.

PMERR_INV_MICROPS_FUNCTION An attempt was made to issue a function that is invalid in a micro presentation space.

Remarks

This function determines whether a new element is to be inserted into a segment, moving any subsequent elements further along the segment, or whether a new element is to replace the current element.

In **SEGEM_INSERT** mode, when an element is generated, it is inserted following the element indicated by the element pointer. The element pointer is updated to point to the new element.

In **SEGEM_REPLACE** mode, when an element is generated, it replaces the element indicated by the element pointer. The element pointer does not change. It is an error if a new element is generated in **SEGEM_REPLACE** mode if the element pointer is 0 (as it is when a segment is opened).

The editing mode can be changed at any time, (except while within an element bracket), and is not an attribute of a specific segment. It only applies to the storing of data within retained segments. It is not an error to issue this function in other drawing modes; the value of the edit mode is set irrespective of the value of the draw mode.

This function is invalid within an element bracket. The default editing mode (set by GpiCreatePS or GpiResetPS) is **SEGEM_INSERT**.

Related Functions

- GpiCreatePS
- GpiOpenSegment
- GpiQueryEditMode

Example Code

This example sets the current editing mode to insert.

```
#define INCL_GPISEGEDITING
#include <OS2.H>

HPS hps;          /* Presentation-space */
                  /* handle. */

GpiSetEditMode(hps, SEGEM_INSERT); /* insert mode. */
```

GpiSetElementPointer – Set Element Pointer

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetElementPointer (HPS hps, LONG IElement)

This function sets the element pointer, within the current segment, to the element number specified.

Parameters

hps (HPS) – input

Presentation-space handle.

IElement (LONG) – input

The element number required.

If the value specified is negative, the element pointer is set to 0.

If the value specified is greater than the number of elements in the segment, the element pointer is set to the last element.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_NOT_IN_RETAIN_MODE

An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to **retain**

PMERR_NO_CURRENT_SEG

An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open segment.

PMERR_INV_MICROPS_FUNCTION

An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_IN_ELEMENT

An attempt was made to issue a function invalid inside an element bracket.

Remarks

The currently open segment has an element pointer that points to a particular element in the segment; each element is placed into the segment at the place indicated by the pointer. When a retained segment is first opened, the element pointer is set to 0 (empty segment). It is incremented each time a call causes an element (a single API call) to be placed in the segment. When a segment is reopened, the element pointer is reset to 0 (that is, before the first element).

The element pointer for a segment is not remembered if the segment is closed and subsequently reopened.

This function is only valid when the drawing mode (see GpiSetDrawingMode) is set to **retain** (not **draw-and-retain**), and a segment bracket is currently in progress. It is invalid within an element bracket.

GpiSetElementPointer — Set Element Pointer

Related Functions

- GpiBeginElement
- GpiDeleteElement
- GpiDeleteElementRange
- GpiDeleteElementsBetweenLabels
- GpiElement
- GpiEndElement
- GpiLabel
- GpiOffsetElementPointer
- GpiQueryElement
- GpiQueryElementPointer
- GpiQueryElementType
- GpiSetElementPointerAtLabel

Example Code

This function sets the element pointer, within the current segment, to 0.

```
#define INCL_GPICONTROL
#define INCL_GPISEGEDITING
#include <OS2.H>

HPS hps;          /* Presentation-space */
                  /* handle. */

/* This example uses the GpiSetElementPointer function to move */
/* the element pointer to an element to be edited. */

GpiSetDrawingMode(hps, DM_RETAIN); /* set DM_RETAIN drawing mode */
GpiOpenSegment(hps, 2L);           /* open segment to edit */
GpiSetElementPointer(hps, 3L);     /* move element pointer
                                   to 3rd element */
GpiSetColor(hps, CLR_GREEN);      /* new element changes
                                   color to green */
GpiCloseSegment(hps);             /* close the segment */
```


GpiSetElementPointerAtLabel — Set Element Pointer At Label

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetElementPointerAtLabel (HPS hps, LONG lLabel)

This function sets the element pointer, within the current segment, to the element containing the specified label.

Parameters

hps (HPS) — input
Presentation-space handle.

lLabel (LONG) — input
Required label.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.
PMERR_NOT_IN_RETAIN_MODE	An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to retain
PMERR_NO_CURRENT_SEG	An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open segment.
PMERR_INV_IN_ELEMENT	An attempt was made to issue a function invalid inside an element bracket.
PMERR_LABEL_NOT_FOUND	The specified element label did not exist.

Remarks

The search starts from the element pointed to by the current element pointer. If the specified label is not found between there and the end of the segment, an error is generated and the element pointer is left unchanged. (Also see GpiSetElementPointer.)

This function is valid only when the drawing mode (see GpiSetDrawingMode) is set to **retain** (not **draw-and-retain**), and a segment bracket is currently open. It is not valid within an element bracket.

GpiSetElementPointerAtLabel – Set Element Pointer At Label

Related Functions

- GpiBeginElement
- GpiDeleteElement
- GpiDeleteElementRange
- GpiDeleteElementsBetweenLabels
- GpiElement
- GpiEndElement
- GpiLabel
- GpiOffsetElementPointer
- GpiQueryElement
- GpiQueryElementPointer
- GpiQueryElementType
- GpiSetElementPointer

Example Code

This function sets the element pointer at label 1.

```
#define INCL_GPISEGEDITING
#include <OS2.H>

HPS hps;          /* Presentation-space */
                  /* handle.          */

GpiSetElementPointerAtLabel(hps, 1L);
```

GpiSetGraphicsField — Set Graphics Field

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetGraphicsField (HPS hps, PRECTL prciField)

This function sets the size and position of the graphics field in presentation page units.

Parameters

hps (HPS) — input
Presentation-space handle.

prciField (PRECTL) — input
Graphics field.

It is an error if the top coordinate is less than the bottom, or the right coordinate is less than the left.

All values are in presentation-page units.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_GRAPHICS_FIELD	An invalid field parameter was specified with GpiSetGraphicsField
PMERR_INV_COORDINATE	An invalid coordinate value was specified.

Remarks

The graphics field specifies a clipping boundary within the presentation page.

The boundaries are inclusive, so that points on them are not clipped (removed). By default, no clipping is performed.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see "Metafile Restrictions" on page G-1.

GpiSetGraphicsField – Set Graphics Field

Related Functions

- GpiQueryDefViewingLimits
- GpiQueryGraphicsField
- GpiQueryViewingLimits
- GpiSetDefViewingLimits
- GpiSetViewingLimits
- GpiExcludeClipRectangle
- GpiIntersectClipRectangle
- GpiOffsetClipRegion
- GpiQueryClipBox
- GpiQueryClipRegion
- GpiSetClipPath
- GpiSetClipRegion

Example Code

This example sets the graphics field to 400 by 400 with the left bottom corner at 25,25.

```
#define INCL_GPITRANSFORMS
#include <OS2.H>

HPS hps;          /* Presentation-space */
                  /* handle.          */

RECTL rc1Field = {25, /* x coordinate of left-hand edge of */
                  /* rectangle. */
                  25, /* y coordinate of bottom edge of */
                  /* rectangle. */
                  425,/* x coordinate of right-hand edge of */
                  /* rectangle. */
                  425};/* y coordinate of top edge of rectangle. */

GpiSetGraphicsField(hps, &rc1Field);
```

GpiSetInitialSegmentAttrs – Set Initial Segment Attributes

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetInitialSegmentAttrs (HPS hps, LONG IAttribute, LONG IValue)

This function specifies a segment attribute that is used when a segment is subsequently created.

Parameters

hps (HPS) – input
Presentation-space handle.

IAttribute (LONG) – input
Segment attribute:

ATTR_DETECTABLE Detectability.

This can be used to determine whether a correlation function can be performed on the primitives within the segment. For correlation on retained segments see:

- GpiCorrelateChain
- GpiCorrelateFrom
- GpiCorrelateSegment.

Correlation on primitives outside segments is controlled by the correlate flag on draw controls (see GpiSetDrawControl).

ATTR_VISIBLE Visibility.

Controls whether a segment is to be made visible on the output medium.

ATTR_CHAINED Chained.

Controls whether the segment is a root segment to be included in the segment drawing chain. In **draw** or **draw-and-retain** modes (see GpiSetDrawingMode) a chained segment is drawn as it passes across the API; an unchained segment is not.

Unchained segments are usually called from another segment. They can also be segments that are inserted into the chain later (with GpiSetSegmentPriority or GpiSetSegmentAttrs), or segments that are drawn individually with GpiDrawSegment.

ATTR_DYNAMIC Dynamic.

Controls whether the segment is to be dynamic; that is, drawn using exclusive-OR, so that it can be readily erased by redrawing it. (See GpiDrawDynamics, GpiRemoveDynamics, and the DCTL_DYNAMIC option of GpiSetDrawControl.)

Only retained segments can be dynamic.

The dynamic segment attribute is always ignored if the segment is not currently chained.

ATTR_FASTCHAIN Fast chaining.

Controls whether, for a chained segment, the system can assume that all primitive attributes need not be reset to default values before execution of the segment.

GpiSetInitialSegmentAttrs – Set Initial Segment Attributes

ATTR_PROP_DETECTABLE Propagate detectability.
Controls whether the value of the detectability attribute for a segment should be propagated (forced) to all segments beneath it in the hierarchy.

ATTR_PROP_VISIBLE Propagate visibility.
Controls whether the value of the visibility attribute for a segment should be propagated (forced) to all segments beneath it in the hierarchy.

IValue (LONG) – input
Attribute value:

ATTR_ON On/yes

ATTR_OFF Off/no.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_ATTR An invalid attribute parameter was specified with GpiSetSegmentAttrs, GpiQuerySegmentAttrs, GpiSetInitialSegmentAttrs, or GpiQueryInitialSegmentAttrs.

PMERR_INV_SEG_ATTR_VALUE An invalid attribute value parameter was specified with GpiSetSegmentAttrs or GpiSetInitialSegmentAttrs.

PMERR_INV_MICROPS_FUNCTION An attempt was made to issue a function that is invalid in a micro presentation space.

Remarks

Initial segment attributes are modal settings used to determine the initial attributes of new segments as they are created; that is, when an GpiOpenSegment function is issued, and the segment does not already exist. The default values of initial segment attributes are:

- Not detectable
- Visible
- Chained
- Not dynamic
- Fast chaining
- Propagate detectability
- Propagate visibility.

A nonretained segment can never be given the **dynamic** attribute.

Primitives outside segments are not affected by GpiSetInitialSegmentAttrs.

GpiSetInitialSegmentAttrs — Set Initial Segment Attributes

Related Functions

- GpiCallSegmentMatrix
- GpiCloseSegment
- GpiCorrelateSegment
- GpiDeleteSegment
- GpiDeleteSegments
- GpiDrawSegment
- GpiErrorSegmentData
- GpiDrawSegment
- GpiQueryInitialSegmentAttrs
- GpiSetSegmentAttrs
- GpiSetSegmentPriority

Example Code

This function specifies a segment attribute that is used when a segment is subsequently created. In this example, the most common attributes are selected.

```
#define INCL_GPISEGMENTS
#include <OS2.H>
```

```
HPS hps;          /* Presentation-space */
                  /* handle.          */
```

```
GpiSetInitialSegmentAttrs (hps,
                           ATTR_DETECTABLE |
                           ATTR_VISIBLE |
                           ATTR_DYNAMIC |
                           ATTR_PROP_DETECTABLE |
                           ATTR_PROP_VISIBLE,
                           ATTR_ON);
```

GpiSetLineEnd – Set Line End

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

```
BOOL GpiSetLineEnd (HPS hps, LONG lLineEnd)
```

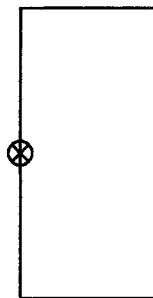
This function sets the current line-end attribute.

Parameters

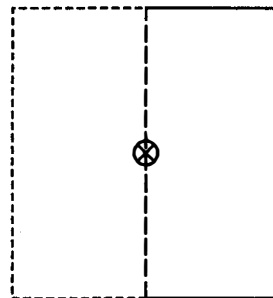
hps (HPS) – input
Presentation-space handle.

lLineEnd (LONG) – input
Style of line end:

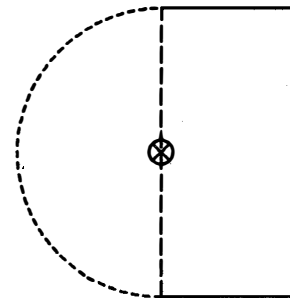
Flat



Square



Round



⊗ Geometric point of line end

----- Outline of end shape

LINEEND_DEFAULT Use default, same as LINEEND_FLAT (unless changed with GpiSetDefAttrs)

LINEEND_FLAT Flat

LINEEND_SQUARE Square

LINEEND_ROUND Round.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LINE_END_ATTR

An invalid line end attribute value was specified.

GpiSetLineEnd — Set Line End

Remarks

The line-end attribute defines the shape of the ends of lines or arcs at the beginning and end of an open figure. This attribute is used only in the GpiModifyPath function (with a *IMode* parameter of MPATH_STROKE) or in the GpiStrokePath function.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the line-end attribute is preserved.

Related Functions

- GpiLine
- GpiPolyLine
- GpiQueryLineEnd
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom

Graphic Elements and Orders

Element Type: **OCODE_GSLE**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Line End**

Element Type: **OCODE_GPSLE**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Line End**

Example Code

This function sets the line end to be square (as opposed to round for example).

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;          /* Presentation-space */
                  /* handle.          */

GpiSetLineEnd(hps,
              LINEEND_SQUARE);
```

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

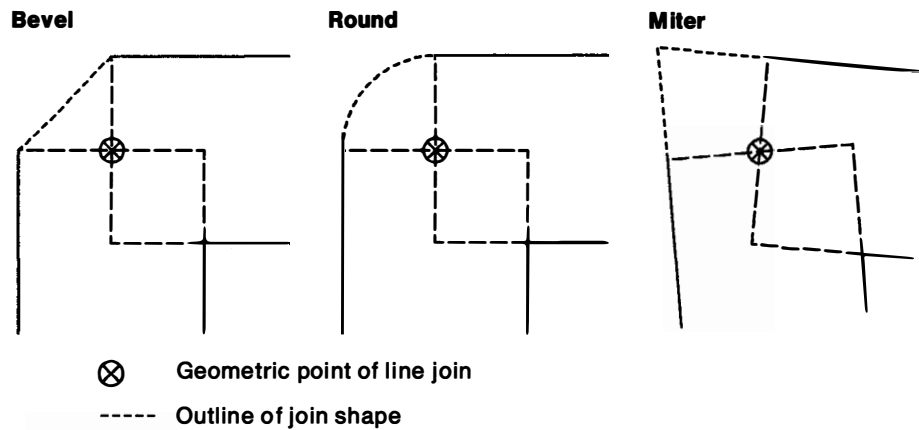
```
BOOL GpiSetLineJoin (HPS hps, LONG iLineJoin)
```

This function sets the current line-join attribute.

Parameters

hps (**HPS**) – input
Presentation-space handle.

iLineJoin (**LONG**) – input
Style of line join:



LINEJOIN_DEFAULT Use default, same as **LINEJOIN_BEVEL** (unless changed with **GpiSetDefAttrs**)

LINEJOIN_BEVEL Bevel

LINEJOIN_ROUND Round

LINEJOIN_MITRE Miter.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from **WinGetLastError**

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LINE_JOIN_ATTR

An invalid line join attribute value was specified.

GpiSetLineJoin —

Set Line Join

Remarks

The line-join attribute defines how individual lines and arcs within a figure are joined together. This attribute is used only during a GpiModifyPath function (with a *IMode* parameter of MPATH_STROKE) or a GpiStrokePath function.

For LINEJOIN_MITRE, where the lines going into a join are nearly parallel (a very sharp change in direction), a miter join could potentially extend to a distance that approaches infinity. To prevent this, whenever the ratio of the miter length to the geometric line width exceeds **10**, a bevel join is drawn instead. (The miter length is the distance from the point at which the inner edges of the wideline intersect, to the point at which the outer edges of the wideline intersect.)

The attribute mode (see GpiSetAttrMode) determines whether the current value of the line-join attribute is preserved.

Related Functions

- GpiLine
- GpiPolyLine
- GpiQueryLineEnd
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetLineEnd
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom

Graphic Elements and Orders

Element Type: **OCODE_GSLJ**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Line Join**

Element Type: **OCODE_GPSLJ**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Line Join**

Example Code

This function sets the line-join to be round (as opposed to bevel or miter).

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;          /* Presentation-space */
                  /* handle.          */

GpiSetLineEnd(hps,
              LINEJOIN_ROUND);
```

GpiSetLineType – Set Line Type

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```









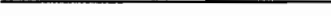
BOOL GpiSetLineType (HPS hps, LONG lLineType)

This function sets the current cosmetic line-type attribute.

Parameters

hps (HPS) – input
Presentation-space handle.

lLineType (LONG) – input
Line types available:

LINETYPE_DEFAULT	- Solid line (the default)	
LINETYPE_DOT	- Dotted line	
LINETYPE_SHORTDASH	- Short-dashed line	
LINETYPE_DASHDOT	- Dash-dot line	
LINETYPE_DOUBLEDOT	- Double-dotted line	
LINETYPE_LONGDASH	- Long-dashed line	
LINETYPE_DASHDOUBLEDOT	- Dash-double-dot line	
LINETYPE_SOLID	- Solid line	
LINETYPE_ALTERNATE	- Alternate pels on	
LINETYPE_INVISIBLE	- Invisible line	

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_LINE_TYPE_ATTR	An invalid line type attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

Remarks

A nonsolid line type consists of a sequence of “on” and “off” runs of pels that gives the appearance of a dotted or a dashed line, for example.

This attribute specifies the cosmetic line type, which is used for all line and curve drawing. It does not depend upon transforms, so that, for example, dashes do not become longer when a “zoom in” occurs.

GpiSetLineType — Set Line Type

The standard line types are implemented on each device to give a good appearance on that device, taking into account the pel resolution. Their definitions cannot be changed by applications, nor may applications define additional cosmetic line types.

The system maintains position within the line-type definition so that, for example, a curve may be implemented as a polyline. However, some functions cause position to be reset to the start of the definition. These are:

- GpiCallSegmentMatrix
- GpiMove
- GpiPop (or end of called segment) that pops current position or a model transform
- GpiSetCurrentPosition
- GpiSetLineType
- GpiSetModelTransformMatrix
- GpiSetPageViewport
- GpiSetSegmentTransformMatrix.

The default line-type is solid. This can be changed with GpiSetDefAttrs.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the line-type attribute is preserved.

Related Functions

- GpiBox
- GpiLine
- GpiPolyLine
- GpiQueryLineEnd
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineWidth
- GpiSetLineWidthGeom

Graphic Elements and Orders

Element Type: **OCODE_GSLT**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Line Type**

Element Type: **OCODE_GPSLT**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Line Type**

GpiSetLineType – Set Line Type

Example Code

This function sets the line-type to be round (as opposed to bevel or miter).

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
```

```
HPS hps;          /* Presentation-space */
                  /* handle.           */
```

```
GpiSetLineType(hps,
                LINETYPE_DEFAULT);
```

GpiSetLineWidth — Set Line Width

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetLineWidth (HPS hps, FIXED fxLineWidth)

This function sets the current cosmetic line-width attribute.

Parameters

hps (HPS) — input

Presentation-space handle.

fxLineWidth (FIXED) — input

Line-width multiplier

LINEWIDTH_DEFAULT Use default; same as LINEWIDTH_NORMAL (unless changed with GpiSetDefAttrs).

LINEWIDTH_NORMAL Normal width (1.0).

Any other positive value is a multiplier on the "normal" line width.

LINEWIDTH_THICK Thick.

Where only two line thicknesses, "normal" and "thick," are supported, "normal" will be used for values less than or equal to 1.0 (other than LINEWIDTH_DEFAULT), and "thick" otherwise.

See DevQueryCaps (CAPS_ADDITIONAL_GRAPHICS and CAPS_LINEWIDTH_THICK).

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LINE_WIDTH_ATTR An invalid line width attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_UNSUPPORTED_ATTR_VALUE An attribute value was specified with GpiSetAttrs that is not supported.

Remarks

The cosmetic line width specifies a multiplier on the "normal" line thickness for the device. Cosmetic thickness does not depend upon transforms, so that, for example, lines do not become thicker when a "zoom-in" occurs.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the line-width attribute is preserved.

GpiSetLineWidth — Set Line Width

Related Functions

- GpiBox
- GpiLine
- GpiPolyLine
- GpiQueryLineEnd
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidthGeom

Graphic Elements and Orders

Element Type: **OCODE_GSFLW**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Fractional Line Width**

Element Type: **OCODE_GPSFLW**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Fractional Line Width**

Example Code

This function sets the line width to the default, so that there is no multiplying factor.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
```

```
HPS hps;          /* Presentation-space */
                  /* handle.                */
```

```
GpiSetLineWidth(hps,
                LINEWIDTH_NORMAL);
```


GpiSetLineWidthGeom — Set Line Width Geom

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetLineWidthGeom (HPS hps, LONG ILineWidth)

This function sets the current geometric line-width attribute.

Parameters

hps (HPS) — input
Presentation-space handle.

ILineWidth (LONG) — input
Geometric line width.

The geometric line width in world coordinates. It must not be negative.

A thickness of 0 results in an area of 0 width. Because filling includes the boundaries, this results in the thinnest possible lines and arcs, regardless of what transforms are in force.

The initial default value of the geometric line width is 1. This can be changed with GpiSetDefAttrs.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_GEOM_LINE_WIDTH_ATTR An invalid geometric line width attribute value was specified.

Remarks

The geometric line-width attribute is used only in the GpiModifyPath function (with a *IMode* of MPATH_STROKE) or in the GpiStrokePath function. This attribute specifies the width to be used in converting the lines and arcs, of which the path is composed, into wide lines and arcs. The resulting shape is treated like an area, so the boundaries are considered to be part of its interior. This means that the width of the lines and arcs is one pel wider than the geometric line width transformed to device coordinates.

The geometric line width is specified in world-coordinate units, so that, for example, the thickness varies on a zoom operation.

Normal line and curve drawing uses only the cosmetic line width (see GpiSetLineWidth).

This function must not be issued within an area or path bracket.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the geometric line width is preserved.

GpiSetLineWidthGeom — Set Line Width Geom

Related Functions

- GpiLine
- GpiPolyLine
- GpiQueryLineEnd
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiBeginPath
- GpiCloseFigure
- GpiEndPath
- GpiFillPath
- GpiModifyPath
- GpiOutlinePath
- GpiPathToRegion
- GpiSetClipPath
- GpiStrokePath

Graphic Elements and Orders

Element Type: **OCODE_GSSLW**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Stroke Line Width**

Element Type: **OCODE_GPSSLW**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Stroke Line Width**

Example Code

This function sets the line width geometry to double the default of 1.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;          /* Presentation-space */
                  /* handle. */

GpiSetLineWidthGeom(hps,
                    2L);
```

GpiSetMarker — Set Marker

#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */

BOOL GpiSetMarker (HPS hps, LONG ISymbol)

This function sets the value of the marker-symbol attribute.

Parameters

hps (HPS) — input
Presentation-space handle.

ISymbol (LONG) — input
Marker symbol.

The identity of the required marker symbol. Zero selects the default marker symbol, a value in the range 1 through 255 identifies a symbol in the current marker set. Valid values in the default marker set are shown below, these symbols are not necessarily available with other marker sets:

MARKSYM_DEFAULT	The default; same as MARKSYM_CROSS
MARKSYM_CROSS	×
MARKSYM_PLUS	+
MARKSYM_DIAMOND	◇
MARKSYM_SQUARE	□
MARKSYM_SIXPOINTSTAR	✱
MARKSYM_EIGHTPOINTSTAR	✳
MARKSYM_SOLIDDIAMOND	◆
MARKSYM_SOLIDSQUARE	■
MARKSYM_DOT	•
MARKSYM_SMALLCIRCLE	○
MARKSYM_BLANK	(blank)

Returns

Success indicator:

- TRUE** Successful completion
- FALSE** Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_MARKER_SYMBOL_ATTR	An invalid marker symbol attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

Remarks

This function must not be issued in an area bracket.

The default marker-symbol is a cross. This can be changed with GpiSetDefAttrs.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the marker attribute is to be preserved.

Related Functions

- GpiMarker
- GpiPolyMarker
- GpiQueryMarker
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMarkerBox
- GpiSetMarkerSet
- GpiSetMix

Graphic Elements and Orders

Element Type: **OCODE_GSMT**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Marker Symbol**

Element Type: **OCODE_GPSMT**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Marker Symbol**

Example Code

This function changes the marker from the default (a cross) to a diamond.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;          /* Presentation-space */
                  /* handle.          */

GpiSetMarker(hps,
             MARKSYM_DIAMOND);
```

GpiSetMarkerBox — Set Marker Box

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetMarkerBox (HPS hps, PSIZEF pszfxSize)

This function sets the current marker-box attribute.

Parameters

hps (HPS) — input

Presentation-space handle.

pszfxSize (PSIZEF) — input

Size of marker box.

The size is specified in world coordinates. The fractional part of the value should be 0.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

Remarks

The value of the marker-box attribute affects the size of markers that are selected from a vector font only. The size of markers that are selected from an image font is not affected by this attribute.

For default markers, this attribute only has an effect if the device supports the scaling of default markers, that is, the CAPS_SCALED_DEFAULT_MARKERS parameter in the CAPS_ADDITIONAL_GRAPHICS element of the device capabilities array returned by the DevQueryCaps function is set to 1.

This function must not be issued in an area bracket.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the marker-box attribute is preserved.

The initial default value of the marker box is the size returned by DevQueryCaps (CAPS_MARKER_WIDTH and CAPS_MARKER_HEIGHT), for the currently associated device, converted to presentation page space.

The default value can be changed with GpiSetDefAttrs.

Related Functions

- DevQueryCaps
- GpiMarker
- GpiPolyMarker
- GpiQueryMarkerBox
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetDefAttrs
- GpiSetMarker
- GpiSetMarkerSet
- GpiSetMix

Graphic Elements and Orders

Element Type: **OCODE_GSMC**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Marker Cell**

Element Type: **OCODE_GPSMC**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Marker Cell**

Example Code

This function sets the marker box to 10 by 10.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;          /* Presentation-space */
                  /* handle. */

SIZEF fxSize = {MAKEFIXED(10,0),
                MAKEFIXED(10,0)};
                  /* The size is specified in */
                  /* world coordinates. The */
                  /* fractional part of the */
                  /* value should be zero. */

GpiSetMarkerBox(hps,
                &fxSize);
```

GpiSetMarkerSet — Set Marker Set

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetMarkerSet (HPS hps, LONG ISet)

This function sets the current marker-set attribute.

Parameters

hps (HPS) — input

Presentation-space handle.

ISet (LONG) — input

Marker-set local identifier.

The identity (Icid) of the required marker set:

LCID_DEFAULT Default (can be set explicitly with GpiSetDefAttrs)

1 – 254 Identifies a logical font.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MARKER_SET_ATTR

An invalid marker set attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_HUGE_FONTS_NOT_SUPPORTED

An attempt was made using GpiSetCharSet, GpiSetPatternSet, GpiSetMarkerSet, or GpiSetAttrs to select a font that is larger than the maximum size (64Kb) supported by the target device driver.

Remarks

This function must not be issued in an area bracket.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the marker-set attribute is preserved.

If the default marker set is changed (using GpiSetDefAttrs) the initial default marker set cannot be selected with GpiSetMarkerSet.

Related Functions

- GpiMarker
- GpiPolyMarker
- GpiQueryMarkerSet
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetMarker
- GpiSetMarkerBox

Graphic Elements and Orders

Element Type: **OCODE_GSMS**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Marker Set**

Element Type: **OCODE_GPSMS**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Marker Set**

Example Code

This function changes the marker set to one defined by the logical font with id 26.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
```

```
HPS hps;          /* Presentation-space */
                  /* handle.          */
```

```
GpiSetMarkerSet(hps,
                26L);
```


GpiSetMetaFileBits — Set Metafile Bits

```
#define INCL_GPIMETAFILES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetMetaFileBits (HMF hmf, LONG IOffset, LONG ILength, PBYTE pbBuffer)

This call transfers metafile data from application storage into a memory metafile.

Parameters

hmf (HMF) — input

Metafile-memory handle.

IOffset (LONG) — input

Offset.

Offset, in bytes, into the metafile data from where the transfer must start. This is used when the metafile data is too long to fit into a single application buffer.

ILength (LONG) — input

Length of the metafile data.

pbBuffer (PBYTE) — input

Metafile data buffer.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HMF

An invalid metafile handle was specified.

PMERR_INV_METAFILE_LENGTH

An invalid length parameter was specified with GpiSetMetaFileBits or GpiQueryMetaFileBits.

PMERR_INV_METAFILE_OFFSET

An invalid length parameter was specified with GpiSetMetaFileBits or GpiQueryMetaFileBits.

PMERR_METAFILE_IN_USE

An attempt has been made to access a metafile that is in use by another thread.

Remarks

The application must ensure that the data is in the correct format. It should not have been changed since it was created by GpiQueryMetaFileBits.

The length of the metafile is increased, if necessary, to accommodate the supplied data. If the supplied data is shorter, the metafile length is not reduced. However, in this case the metafile is still valid, if the data in it is complete and otherwise correct.

GpiSetMetaFileBits – Set Metafile Bits

Related Functions

- GpiCopyMetaFile
- GpiDeleteMetaFile
- GpiLoadMetaFile
- GpiPlayMetaFile
- GpiQueryMetaFileBits
- GpiQueryMetaFileLength
- GpiSaveMetaFile

Example Code

This example shows how to copy a metafile into application storage to edit the contents and then write back to the metafile using the GpiSetMetaFileBits call.

```
#define INCL_GPIMETAFILES
#include <OS2.H>

HPS hps;          /* Presentation-space */
                  /* handle. */

HMF hmf;
PBYTE pbBuffer;
LONG cBytes;
LONG lOffset;

hmf = GpiLoadMetaFile(hps, "sample.met");

/* Allocate the buffer for the metafile data. */

cBytes = GpiQueryMetaFileLength(hmf); /* gets length of metafile */

DosAllocMem((PPVOID)pbBuffer,
            cBytes,
            PAG_READ |
            PAG_WRITE |
            PAG_COMMIT);

GpiQueryMetaFileBits(
    hmf,          /* handle of metafile */
    lOffset,      /* offset of next byte to retrieve */
    cBytes,       /* retrieves cBytes */
    pbBuffer);    /* buffer to receive metafile data */

/* . */
/* work with the metafile */
/* . */

/* write data back to the metafile */

GpiSetMetaFileBits(hmf,
                   lOffset,
                   cBytes,
                   pbBuffer);
```

GpiSetMix – Set Mix

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetMix (HPS hps, LONG IMixMode)

This function sets the current foreground mix attribute for each individual primitive type.

Parameters

hps (HPS) – input
Presentation-space handle.

IMixMode (LONG) – input
Mix mode.

Defines the color-mixing mode.

Mixing other than FM_LEAVEALONE or FM_OVERPAINT is done on the physical color index. In general, this corresponds to the color index of the logical color table if an indexed color table has been realized. In other circumstances, the color that results from such a mix cannot be predicted. Nevertheless, if FM_XOR is supported for example, drawing the same object twice with a foreground mix of FM_XOR and a background mix of BM_LEAVEALONE with no intervening drawing in other mix modes, causes the object to be erased cleanly.

The currently associated device supports any of the mixes specified as supported in DevQueryCaps (CAPS_FOREGROUND_MIX_SUPPORT). Any other valid mixes may be supported for some primitive types, but otherwise results in FM_OVERPAINT. An error is raised only if the value specified is not one of those listed below.

Note: Mixes marked with an asterisk (*) are mandatory for all devices, except that FM_OR is only mandatory for devices capable of supporting it. FM_XOR is mandatory only on displays.

FM_DEFAULT	Use default, the same as FM_OVERPAINT, unless changed with GpiSetDefAttrs
FM_OR	Logical-OR (*)
FM_OVERPAINT	Overpaint (*)
FM_XOR	Logical-XOR (*)
FM_LEAVEALONE	Leave alone (invisible) (*)
FM_AND	Logical-AND
FM_SUBTRACT	(Inverse source) AND destination
FM_MASKSRCNOT	Source AND (inverse destination)
FM_ZERO	All zeros
FM_NOTMERGESRC	Inverse (source OR destination)
FM_NOTXORSRC	Inverse (source XOR destination)
FM_INVERT	Inverse (destination)
FM_MERGESRCNOT	Source OR (inverse destination)
FM_NOTCOPYSRC	Inverse (source)
FM_MERGENOTSRC	(Inverse source) OR destination
FM_NOTMASKSRC	Inverse (source AND destination)
FM_ONE	All ones.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MIX_ATTR

An invalid mix attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

Remarks

The current values for each primitive type are updated. The attribute mode (see GpiSetAttrMode) determines whether the current value of the mix attribute is preserved.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see “Metafile Restrictions” on page G-1.

Related Functions

- DevQueryCaps
- GpiBeginArea
- GpiBox
- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiEndArea
- GpiFullArc
- GpiLine
- GpiMarker
- GpiMove
- GpiPartialArc
- GpiPointArc
- GpiPolyFillet
- GpiPolyFilletSharp
- GpiPolyLine
- GpiPolyMarker
- GpiPolySpline
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt
- GpiQueryMix
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetDefAttrs
- WinSetSysColors

GpiSetMix — Set Mix

Graphic Elements and Orders

Element Type: **OCODE_GSMX**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Mix**

Element Type: **OCODE_GPSMX**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Mix**

Example Code

This function sets the current foreground mix attribute for each individual primitive type.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;          /* Presentation-space */
                  /* handle.          */

GpiSetMix(hps,
          FM_LEAVEALONE);
```

GpiSetModelTransformMatrix – Set Model Transform Matrix

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
```

```
BOOL GpiSetModelTransformMatrix (HPS hps, LONG ICount, PMATRIXLF pmatlfArray,  
                                LONG IOptions)
```

This function sets the model transform matrix for subsequent primitives.

Parameters

hps (**HPS**) – input
Presentation-space handle.

ICount (**LONG**) – input
Number of elements in matrix.

The number of elements of *pmatlfArray* to be examined, starting from the beginning of the structure. If *ICount* is less than 9, remaining elements default to the corresponding elements of the identity matrix. If *ICount* = 0, the identity matrix is used.

pmatlfArray (**PMATRIXLF**) – input
Transformation matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type **FIXED**, and have an assumed binary point between the second and third bytes. Thus a value of 1.0 is represented by 65 536. Other elements are normal signed integers. If the presentation space coordinate format is **GPIF_SHORT** (see **GpiCreatePS**), these elements must be within the range –1 through +1.

The third, sixth, and ninth elements, when specified, must be 0, 0, and 1, respectively.

IOptions (**LONG**) – input
Transform options.

Specifies how the transform defined by the *pmatlfArray* should be used to modify the existing current model transform (the existing transform is the concatenation, in the current call context, of the instance, segment and model transforms, from the root segment downwards). Possible values are:

- | | |
|--------------------------|--|
| TRANSFORM_REPLACE | The previous model transform is discarded and replaced by the specified transform. |
| TRANSFORM_ADD | The specified transform is combined with the existing model transform, in the order (1) existing transform, (2) new transform. This option is most useful for incremental updates to transforms. |
| TRANSFORM_PREEMPT | The specified transform is combined with the existing model transform, in the order (1) new transform, (2) existing transform. |

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from **WinGetLastError**

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

GpiSetModelTransformMatrix — Set Model Transform Matrix

PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_INV_MATRIX_ELEMENT	An invalid transformation matrix element was specified.
PMERR_INV_TRANSFORM_TYPE	An invalid options parameter was specified with a transform matrix function.

Remarks

The matrix is used to update the previous current model transform, depending upon the value of *IOptions*.

The transform is specified as a one-dimensional array of *ICount* elements, being the first elements of a 3-row by 3-column matrix ordered by rows. The order of the elements is:

Matrix	Array
$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix}$	(a,b,0,c,d,0,e,f,1)

The transform acts on the coordinates of the primitives in a segment, so that a point with coordinates (x,y) is transformed to the point:

$(a*x + c*y + e, b*x + d*y + f)$

If scaling values greater than unity are given (which only applies if the presentation space coordinate format as set by the GpiCreatePS function is GPIF_LONG) it is possible for the combined effect of this, and any other relevant transforms, to exceed fixed-point implementation limits. This causes an error.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the model transform is preserved.

Model transforms can apply to primitives either inside or outside segments.

Related Functions

- GpiCallSegmentMatrix
- GpiQueryModelTransformMatrix
- GpiQuerySegmentTransformMatrix
- GpiSetSegmentTransformMatrix

Graphic Elements and Orders

Element Type: **OCODE_GSTM**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Model Transform**

Element Type: **OCODE_GPSTM**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Model Transform**

GpiSetModelTransformMatrix — Set Model Transform Matrix

Example Code

This function sets the model transformation matrix as one which scales everything by a factor of 2.

```
#define INCL_GPITRANSFORMS
#include <OS2.H>

HPS hps;          /* Presentation-space */
                  /* handle. */
MATRIXLF matlf = { MAKEFIXED(2,0), /* see pmgpi.h for a */
                  /* definition of the */
                  /* MAKEFIXED macro. */
                  0, 0, 0,
                  MAKEFIXED(2,0),
                  0, 0, 0, 1};

GpiSetModelTransformMatrix(hps,
                           1L,
                           &matlf,
                           TRANSFORM_REPLACE);
```


GpiSetPageViewport – Set Page Viewport

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetPageViewport (HPS hps, PRECTL prclViewport)

This function sets the page viewport within device space.

Parameters

hps (HPS) – input
Presentation-space handle.

prclViewport (PRECTL) – input
Page viewport.

The page viewport is specified in device units.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_PAGE_VIEWPORT	An invalid viewport parameter was specified with GpiSetPageViewport.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.

Remarks

The presentation page maps to the page viewport and together they define the device transform.

When a presentation space is associated with a device context, a default page viewport is set up.

The origin in device space is mapped to the bottom-left of the output media (window or paper, for example).

This function must not be issued when there is no device context associated with the presentation space.

This function is ignored if issued to a presentation space that is associated with a device context of type OD_QUEUED (with PM_Q_STD data), OD_METAFILE, or OD_METAFILE_NOQUERY.

Related Functions

- GpiCreatePS
- GpiQueryPageViewport

GpiSetPageViewport — Set Page Viewport

Example Code

This example sets the area of the device in which the picture is displayed to page viewport within device space.

```
#define INCL_GPITRANSFORMS
#include <OS2.H>
HPS hps;          /* Presentation-space */
                  /* handle. */
RECTL rc1Field = {25L, /* x coordinate of left-hand edge of */
                  /* rectangle. */
                  25L, /* y coordinate of bottom edge of */
                  /* rectangle. */
                  425L, /* x coordinate of right-hand edge of */
                  /* rectangle. */
                  425L}; /* y coordinate of top edge of
                  /* rectangle. */

GpiSetPageViewport(hps, &rc1Field);
```

GpiSetPaletteEntries —

Set Palette Entries

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetPaletteEntries (HPAL hpal, ULONG ulFormat, ULONG ulStart, ULONG ulCount, PULONG aTable)

This function changes the entries in a palette.

Parameters

hpal (HPAL) — input
Palette handle.

ulFormat (ULONG) — input
Format of entries in the table:

LCOLF_CONSECRGB Array of RGB values, corresponding to color indexes *ulStart* upwards.
Each entry is 4 bytes long.

This is currently the only valid value for this parameter.

ulStart (ULONG) — input
Starting index.

ulCount (ULONG) — input
Count of elements in *aTable*.

This must be greater than or equal to 0.

aTable (PULONG) — input
Start of the application data area.

This contains the palette definition data. The format depends on the value of *ulFormat*.

Each color value is a 4-byte integer, with a value of

$(F * 16777216) + (R * 65536) + (G * 256) + B$

where:

F is a flag byte, which can take the following values (these can be ORed together if required):

PC_RESERVED This index is an animating index. This means that the application might frequently change the RGB value, so the system should not map the logical index of the palette of another application to the entry in the physical palette used for this color.

PC_EXPLICIT The low-order word of the logical color table entry designates a physical palette entry. This allows an application to show the contents of the device palette as realized for other logical palettes. This does not prevent the color in the entry from being changed for any reason.

R is red intensity value

G is green intensity value

B is blue intensity value.

The maximum intensity for each primary is 255.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

GpiSetPaletteEntries – Set Palette Entries

Possible returns from WinGetLastError

PMERR_INV_HPAL	An invalid color palette handle was specified.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_INV_COLOR_DATA	Invalid color table definition data was specified with GpiCreateLogColorTable.
PMERR_INV_COLOR_FORMAT	An invalid format parameter was specified with GpiCreateLogColorTable.
PMERR_INV_COLOR_START_INDEX	An invalid starting index parameter was specified with a logical color table or color query function.
PMERR_INSUFFICIENT_MEMORY	The operation terminated through insufficient memory.
PMERR_PALETTE_BUSY	An attempt has been made to reset the owner of a palette when it was busy.
PMERR_INV_IN_AREA	An attempt was made to issue a function invalid inside an area bracket. This can be detected while the actual drawing mode is draw or draw-and-retain or during segment drawing or correlation functions.

Remarks

The changes made by this function do not become apparent until WinRealizePalette is called, even for animating indices. Changes can be made more rapidly using GpiAnimatePalette with animating indices, assuming that the hardware being used supports this.

GpiSetPaletteEntries can be called at any time to change a logical palette, and the physical palette of the device will incorporate the changes as best it can. However, the system cannot guarantee that a change will be realized in the hardware palette, since realization depends on whether the associated window is in the foreground and on the number of available hardware palette entries.

All presentation spaces that have this palette selected into them (see GpiSelectPalette), are updated with the effects of this function.

If a palette is selected into a presentation space that is associated with a device context of type OD_METAFILE or OD_METAFILE_NOQUERY, only the final color values are recorded in the metafile. This means that, while metafileing, this function must only be used for incremental additions to the color table.

It is an error if a palette is selected into a presentation space that is within an area or path definition when this function is issued.

Related Functions

- GpiAnimatePalette
- GpiCreatePalette
- GpiDeletePalette
- GpiQueryPalette
- GpiQueryPaletteInfo
- GpiSelectPalette
- WinRealizePalette

GpiSetPaletteEntries —

Set Palette Entries

Example Code

This example changes the entries in a palette.

```
#define INCL_GPILOGCOLORTABLE
#include <OS2.H>

HPAL hpal; /* palette handle */
UINT R, G, B;
typedef struct ENTRY
{
    ULONG index;
    ULONG pal_def;
}Entry;

struct TABLE
{
    Entry entry1;
    Entry entry2;
    Entry entry3;
}Table;
BYTE F = PC_RESERVED;

/* In our table, there are 3 8-byte entries. The first 4 bytes */
/* of each entry represent the index and the second 4 bytes of */
/* each entry represent the value of the following formula: */
/* (F * 16777216) + (R * 65536) + (G * 256) + B */
/* which is the palette definition. */
/* where F is the flag PC_RESERVED and R,G,B are the red, */
/* green, and blue intensity values respectively. */

F = 10; R = 10; G = 10;
Table.entry1.pal_def = (F * 16777216)+(R * 65536)+(G * 256) + B;
Table.entry1.index = 0L;

F = 25; R = 25; G = 25;
Table.entry2.pal_def = (F * 16777216)+(R * 65536)+(G * 256) + B;
Table.entry2.index = 1L;

F = 40; R = 40; G = 40;
Table.entry3.pal_def = (F * 16777216)+(R * 65536)+(G * 256) + B;
Table.entry3.index = 2L;

GpiSetPaletteEntries(hpal,
    LCOLF_CONSECRGB, /* Array of RGB values, */
    /* corresponding to color */
    /* indexes lStart */
    /* upwards. Each entry */
    /* is 4 bytes long. */
    0L, /* start at zero. */
    3L, /* elements in table. */
    &Table.entry1.index); /* first element in table. */
```

GpiSetPattern – Set Pattern

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

BOOL GpiSetPattern (HPS hps, LONG IPatternSymbol)

This function sets the current value of the pattern-symbol attribute.

Parameters

hps (HPS) – input
Presentation-space handle.

IPatternSymbol (LONG) – input
Pattern symbol.

Identifies the shading pattern to be used to fill areas. The pattern that appears depends on the particular pattern set selected by the pattern-set attribute. A value of 0 selects the default pattern and values in the range 1 through 255 select particular patterns within the set.

Possible values if the default pattern set has been selected are:

Symbolic name	Description	Pattern number (see Figure 5-10)
PATSYM_DEFAULT	The default; same as PATSYM_SOLID (unless changed with GpiSetDefAttrs).	
PATSYM_DENSE1 through PATSYM_DENSE8	Solid shading with decreasing density	1 through 8
PATSYM_VERT	Vertical pattern	9
PATSYM_HORIZ	Horizontal pattern	10
PATSYM_DIAG1	Diagonal pattern 1, bottom left to top right	11
PATSYM_DIAG2	Diagonal pattern 2, bottom left to top right	12
PATSYM_DIAG3	Diagonal pattern 3, top left to bottom right	13
PATSYM_DIAG4	Diagonal pattern 4, top left to bottom right	14
PATSYM_NOSHADE	No shading	15
PATSYM_SOLID	Solid shading	16
PATSYM_HALFTONE	Alternate pels set on	
PATSYM_BLANK	Blank (same as PATSYM_NOSHADE)	

Note: The pattern PATSYM_HALFTONE can be the same as PATSYM_DENSE4. On non bit-mapped devices it may be mapped to another base pattern.

If the specified pattern is not valid, the default (device-dependent) pattern is used.

GpiSetPattern – Set Pattern

Returns

Success indicator:

- TRUE** Successful completion
- FALSE** Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_PATTERN_ATTR	An invalid pattern symbol attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

Remarks

Any symbol from a raster font can be used as a pattern by the appropriate use of this function and the GpiSetPatternSet function.

If the current pattern set specifies a bit map (see GpiSetBitmapId and GpiSetPatternSet), the pattern attribute is ignored.

If *IPatternSymbol* is set or defaulted to PATSYM_SOLID, and the *ISet* parameter of GpiSetPatternSet is LCID_DEFAULT, pattern colors that are not available may be approximated by dithering (unless dithering has been disabled by setting the LCOL_PURECOLOR bit on the *fiOptions* parameter of GpiCreateLogColorTable).

This function must not be issued in an area or path bracket.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the pattern symbol is preserved.

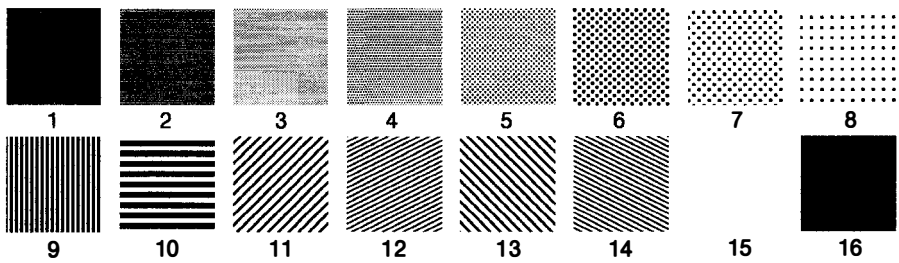


Figure 5-10. Shading patterns in the default pattern set

Related Functions

- GpiBeginArea
- GpiEndArea
- GpiQueryPattern
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetDefAttrs
- GpiSetMix
- GpiSetPatternRefPoint
- GpiSetPatternSet

Graphic Elements and Orders

Element Type: **OCODE_GSPT**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Pattern Symbol**

Element Type: **OCODE_GPSPT**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Pattern Symbol**

Example Code

This function sets the current value of the pattern-symbol to horizontal. This means that when areas are filled, they are filled with a horizontal shading pattern.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
HPS hps;          /* Presentation-space */
                  /* handle. */
```

```
GpiSetPattern(hps,PATSYM_HORIZ);
```


GpiSetPatternRefPoint – Set Pattern Reference Point

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetPatternRefPoint (HPS hps, PPOINTL pptlRefPoint)

This function sets the current pattern reference point to the specified value.

Parameters

hps (HPS) – input
Presentation-space handle.

pptlRefPoint (PPOINTL) – input
Pattern reference point.
The coordinates are world coordinates.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.

Remarks

The pattern reference point is the point to which the origin of the area filling pattern maps. The pattern is mapped into the area to be filled by conceptually replicating the pattern definition in a horizontal and vertical direction.

Because the pattern reference point is subject to all of the transforms, if an area is moved by changing a transform and redrawing, the fill pattern also appears to move, so as to retain its position relative to the area boundaries.

The pattern reference point, which is specified in world coordinates, need not be inside the actual area to be filled. The pattern reference point is not subject to clipping.

This function must not be issued in an area or path bracket.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the pattern reference point is preserved.

The initial default pattern reference point is (0,0). This can be changed with GpiSetDefAttrs.

GpiSetPatternRefPoint – Set Pattern Reference Point

Related Functions

- GpiBeginArea
- GpiEndArea
- GpiQueryPatternRefPoint
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetPattern
- GpiSetPatternSet

Graphic Elements and Orders

Element Type: **OCODE_GSPRP**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Pattern Reference Point**

Element Type: **OCODE_GPSRP**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Pattern Reference Point**

Example Code

This function sets the current pattern reference point to the specified value.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
HPS hps;          /* Presentation-space */
                  /* handle. */
POINTL pt1RefPoint = {0,0};

GpiSetPatternRefPoint(hps, &pt1RefPoint);
```

GpiSetPatternSet — Set Pattern Set

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetPatternSet (HPS hps, LONG ISet)

This function sets the current pattern-set attribute to the specified value.

Parameters

hps (HPS) — input
Presentation-space handle.

ISet (LONG) — input
Pattern-set local identifier:

LCID_DEFAULT Default (can be set explicitly with GpiSetDefAttrs).

1 – 254 Identifies a logical font or a bit map.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_PATTERN_SET_ATTR	An invalid pattern set attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_PATTERN_SET_FONT	An attempt was made to use an unsuitable font as a pattern set.
PMERR_HUGE_FONTS_NOT_SUPPORTED	An attempt was made using GpiSetCharSet, GpiSetPatternSet, GpiSetMarkerSet, or GpiSetAttrs to select a font that is larger than the maximum size (64Kb) supported by the target device driver.

Remarks

The bit map, or character within the font selected, is used for shading. On some devices, a simplified form of the bit map, or character, is used. For example, only a subset such as the first 8 by 8 pels may be used; also on a monochrome device a color bit map is converted to monochrome.

Some fonts are not suitable, and an error is returned if an attempt is made to set them as the current pattern set. These include device fonts that cannot be used for shading, and any kind of raster font for a plotter device.

This function must not be issued in an area or path bracket.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the pattern-set attribute is preserved.

If the default pattern set is changed (using GpiSetDefAttrs), the initial default pattern marker set cannot be selected with GpiSetPatternSet.

Related Functions

- GpiBeginArea
- GpiCreateLogFont
- GpiEndArea
- GpiQueryPatternSet
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetPattern
- GpiSetPatternRefPoint

Graphic Elements and Orders

Element Type: **OCODE_GSPS**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Pattern Set**

Element Type: **OCODE_GPSPS**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Pattern Set**

Example Code

This function sets the current pattern-set attribute to the logical font with id 35.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
HPS hps;          /* Presentation-space */
                  /* handle.          */
```

```
GpiSetPatternSet(hps, 35L);
```

GpiSetPel — Set Pel

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiSetPel (HPS hps, PPOINTL pptlPoint)

This function sets a pel, at a position specified in world coordinates, using the current (line) color and mix.

Parameters

hps (HPS) — input
Presentation-space handle.

pptlPoint (PPOINTL) — input
Position in world coordinates.

Returns

Correlation and error indicators:

GPI_OK Successful

GPI_HITS Correlate hits

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.

Remarks

This function is subject to all the usual clipping (clip path, clip region, viewing limits, graphics field, visible region), and no error is returned if the point is subject to clipping.

This function is independent of drawing mode (see GpiSetDrawingMode); the effect always occurs immediately, and it is not retained even if the drawing mode is **draw-and-retain** or **retain**. (Its effect is, however, recorded in a metafile, but note that this is only successful if the metafile is replayed on a similar device, with **draw** drawing mode.)

Note: This function must not be used when creating SAA-conforming metafiles; see "Metafile Restrictions" on page G-1.

Related Functions

- DevQueryCaps
- GpiQueryPel
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetDefAttrs
- GpiSetMix

Example Code

This function sets a pel, at a position specified in world coordinates, using the current (line) color and mix.

```
#define INCL_GPIBITMAPS
#include <OS2.H>
HPS hps;          /* Presentation-space */
                  /* handle.          */
```

```
POINTL pt1Point = {0,0};
```

```
GpiSetPel(hps, &pt1Point);
```

GpiSetPickAperturePosition – Set Pick-Aperture Position

```
#define INCL_GPICORRELATION /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetPickAperturePosition (HPS hps, PPOINTL pptlPick)

This function sets the center of the pick aperture, in presentation page space, for subsequent nonretained correlation operations.

Parameters

hps (HPS) – input
Presentation-space handle.

pptlPick (PPOINTL) – input
Center of the pick aperture.
The center is in presentation page coordinates.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.

Related Functions

- GpiQueryPickAperturePosition
- GpiQueryPickApertureSize
- GpiSetPickApertureSize

Example Code

In this example we query the position of the center of the pick aperture.

```
#define INCL_GPICORRELATION
#include <OS2.H>

BOOL flResult;
HPS hps; /* Presentation space handle. */
POINTL ptlPoint = {50L, 50L}; /* Pick-aperture position. */

flResult = GpiSetPickAperturePosition(hps, &ptlPoint);
```

GpiSetPickApertureSize — Set Pick-Aperture Size

```
#define INCL_GPICORRELATION /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetPickApertureSize (HPS hps, LONG IOptions, PSIZEL pszISize)

This function sets the pick-aperture size.

Parameters

hps (HPS) — input
Presentation-space handle.

IOptions (LONG) — input
Setting option:

PICKAP_DEFAULT Use the default pick aperture. The value of *pszISize* is ignored.

PICKAP_REC Use the values specified by *pszISize*.

pszISize (PSIZEL) — input
Pick aperture size.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PICK_APERTURE_OPTION An invalid options parameter was specified with GpiSetPickApertureSize

PMERR_INV_PICK_APERTURE_SIZE An invalid size parameter was specified with GpiSetPickApertureSize

Remarks

The pick aperture can be set either to the default value, or to a specified size in presentation page space. This is used in any subsequent nonretained or retained correlation operations.

The default size is a rectangle in presentation page space that produces a square on the device, with side equal to the default character cell height.

Related Functions

- GpiQueryPickApertureSize
- GpiSetPickAperturePosition
- GpiQueryPickAperturePosition

GpiSetPickApertureSize — Set Pick-Aperture Size

Example Code

In this example we set the pick-aperture size to a 4 by 4 box in world coordinates.

```
#define INCL_GPICORRELATION
#include <OS2.H>

HPS hps;          /* Presentation space handle. */
SIZEL size1;      /* Pick-aperture position.    */

size1.cx = 4L; size1.cy = 4L;
GpiSetQueryPickApertureSize(hps, &size1);
```

GpiSetPS — Set Presentation Space

```
#define INCL_GPICONTROL /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetPS (HPS hps, PSIZEL pszlszsize, ULONG flOptions)

This function sets the presentation space size, units, and format.

Parameters

hps (HPS) — input
Presentation-space handle.

pszlszsize (PSIZEL) — input
Presentation-space size.

flOptions (ULONG) — input
Options.

This contains fields of option bits. For each field, one value should be selected (unless the default is suitable). These values can then be ORed together to generate the parameter.

PS_UNITS

Presentation page size units.

Indicates the units for the presentation page size. In each case, the origin is at the bottom left. Possible values are:

PU_ARBITRARY	Application-convenient units
PU_PELS	Pel coordinates
PU_LOMETRIC	Units of 0.1 mm
PU_HIMETRIC	Units of 0.01 mm
PU_LOENGLISH	Units of 0.01 inch
PU_HIENGLISH	Units of 0.001 inch
PU_TWIPS	Units of 1/1440 inch.

PS_FORMAT

Coordinate format.

Indicates options to be used when storing coordinate values internally in the segment store.

For most calls, the format is not directly visible to an application. However, it is visible during editing (for example, GpiQueryElement). The format also has an effect on the amount of storage required for segment store.

One of these can be selected, for a GPIT_NORMAL presentation space (for a GPIT_MICRO presentation space, only GPIF_DEFAULT is allowed):

GPIF_DEFAULT	Default local format (same as GPIF_LONG)
GPIF_SHORT	2-byte integers
GPIF_LONG	4-byte integers.

PS_TYPE

Presentation space.

This option is ignored.

PS_MODE

Mode.

This option is ignored.

GpiSetPS —

Set Presentation Space

PS_ASSOCIATE

Association indicator.

This option is ignored.

PS_NORESET

Inhibit full reset indicator.

Inhibits the full reset of the presentation space. If this flag is set, a reset equivalent to GRES_SEGMENTS is performed. If it is not set, a full reset (GRES_ALL) is performed.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HDC

An invalid device-context handle or (micro presentation space) presentation-space handle was specified.

PMERR_INV_PS_SIZE

An invalid size parameter was specified with GpiCreatePS or GpiSetPS.

PMERR_INV_OR_INCOMPAT_OPTIONS

An invalid or incompatible (with micro presentation space) options parameter was specified with GpiCreatePS or GpiSetPS.

PMERR_INV_FOR_THIS_DC_TYPE

An attempt has been made to issue GpiRemoveDynamics or GpiDrawDynamics to a presentation space associated with a metafile device context.

Remarks

The presentation space is re-initialized to the same state that occurs as if it had been created using the specified size and option values. However, whether the presentation space is a micro presentation space or a normal presentation space cannot be changed, and any device context that is already associated remains associated.

The presentation space code page is set to the current process code page.

On completion, the presentation space is reset with the equivalent of GRES_ALL (see GpiResetPS), unless PS_NORESET is specified, in which case only the equivalent of a GRES_SEGMENTS reset is performed.

This function cannot be used to a presentation space that is associated with a device context of type OD_QUEUED, OD_METAFILE, or OD_METAFILE_NOQUERY.

Related Functions

- GpiAssociate
- GpiCreatePS
- GpiDestroyPS
- GpiQueryDevice
- GpiQueryPS
- GpiResetPS
- GpiRestorePS
- GpiSavePS

GpiSetPS – Set Presentation Space

Example Code

This function is used to reset the presentation space.

```
#include <OS2.H>
#define INCL_GPICONTROL

HPS hps;          /* presentation space handle */
ULONG f1Options; /* reset options */

f1Options = PU_ARBITRARY | /* arbitrary units. */
           GPIF_DEFAULT | /* normal ps format. */

GpiSetPS(hps, f1Options);
```

GpiSetRegion — Set Region

```
#define INCL_GPIREGIONS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetRegion (HPS hps, HRGN hrgn, LONG lcount, PRECTL arclRectangles)

This function changes a region to be the logical-OR of a set of rectangles.

Parameters

hps (HPS) — input

Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

hrgn (HRGN) — input

Region handle.

lcount (LONG) — input

Count of rectangles.

This is the number of rectangles specified in *arclRectangles*. If *lcount* = 0, the region is set to EMPTY, and *arclRectangles* is ignored.

arclRectangles (PRECTL) — input

Array of rectangles.

The rectangles are specified in device coordinates.

For each rectangle in the array, the value of *xright* must be greater than (or equal to) *xleft*, and *ypop* must be greater than (or equal to) *ybottom*.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

PMERR_INV_HRGN

An invalid region handle was specified.

PMERR_INV_COORDINATE

An invalid coordinate value was specified.

PMERR_INV_RECT

An invalid rectangle parameter was specified.

PMERR_REGION_IS_CLIP_REGION

An attempt was made to perform a region operation on a region that is selected as a clip region.

PMERR_HRGN_BUSY

An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

Remarks

This function is similar to GpiCreateRegion, except that it changes an already existing region to be the logical-OR of the supplied rectangles, instead of creating a new region.

The previous contents of the region are irrelevant. Points on the right-hand and top boundaries are not included in the changed region; points on the left-hand and bottom boundaries, that are not also on the right-hand or top boundaries, (that is, the top-left and bottom-right corner points) are included.

It is invalid if the specified region is currently selected as the clip region (by GpiSetClipRegion).

Related Functions

- GpiCombineRegion
- GpiCreateRegion
- GpiDestroyRegion
- GpiEqualRegion
- GpiOffsetRegion
- GpiPaintRegion
- GpiPtInRegion
- GpiQueryRegionBox
- GpiQueryRegionRects
- GpiRectInRegion

Example Code

In this example we change the region to be the logical-or of a set of rectangles.

```
#define INCL_GPIREGIONS
#include <OS2.H>
#define maxrects 2

BOOL flResult;      /* success indicator. */
HPS hps;            /* presentation space handle. */
HRGN hrgn;          /* region handle. */
RECTL arc1Rect[maxrects] = {{20L, 20L,
                             40L, 40L},
                             {40L, 20L,
                             60L, 40L}};

/* array of rectangle structures */

flResult = GpiSetRegion(hps,
                       hrgn,
                       (LONG)maxrects,
                       /* array of two rectangles. */
                       arc1Rect);
```

GpiSetSegmentAttrs – Set Segment Attributes

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetSegmentAttrs (HPS hps, LONG ISegId, LONG IAttribute, LONG IValue)

This function sets a segment attribute.

Parameters

hps (HPS) – input

Presentation-space handle.

ISegId (LONG) – input

Segment identifier.

The identifier of the segment whose attribute is to be updated. It must be greater than zero.

IAttribute (LONG) – input

Segment attribute.

For details of the following attributes, see the GpiSetInitialSegmentAttrs function.

ATTR_DETECTABLE Detectability

ATTR_VISIBLE Visibility

ATTR_CHAINED Chained

ATTR_DYNAMIC Dynamic

ATTR_FASTCHAIN Fast chaining

ATTR_PROP_DETECTABLE Propagate detectability

ATTR_PROP_VISIBLE Propagate visibility.

IValue (LONG) – input

Attribute value:

ATTR_ON On/yes

ATTR_OFF Off/no.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME

An invalid segment identifier was specified.

PMERR_INV_SEG_ATTR

An invalid attribute parameter was specified with GpiSetSegmentAttrs, GpiQuerySegmentAttrs, GpiSetInitialSegmentAttrs, or GpiQueryInitialSegmentAttrs.

PMERR_INV_SEG_ATTR_VALUE

An invalid attribute value parameter was specified with GpiSetSegmentAttrs or GpiSetInitialSegmentAttrs.

GpiSetSegmentAttrs – Set Segment Attributes

PMERR_SEG_NOT_FOUND

The specified segment identifier did not exist

PMERR_INV_MICROPS_FUNCTION

An attempt was made to issue a function that is invalid in a micro presentation space.

Remarks

This function sets the value of one segment attribute for the specified segment. The segment can be any retained segment.

If the identifier is that of the currently-open segment:

- In **retain** mode, this is valid.
- In **draw-and-retain** mode, the retained segment is updated, but there is no change to the immediate drawing.
- In **draw** mode, it is invalid.

(For a description of drawing mode, see GpiSetDrawingMode).

When a segment is modified from nonchained to chained, it is added to the end of the drawing chain.

Related Functions

- GpiCallSegmentMatrix
- GpiCloseSegment
- GpiCorrelateSegment
- GpiDeleteSegment
- GpiDeleteSegments
- GpiDrawSegment
- GpiErrorSegmentData
- GpiOpenSegment
- GpiQuerySegmentAttrs
- GpiSetInitialSegmentAttrs
- GpiSetSegmentPriority

GpiSetSegmentAttrs — Set Segment Attributes

Example Code

This function is used to set the current value of the specified attribute.

```
#define INCL_GPISEGMENTS
#include <OS2.H>

HPS hps; /* Presentation-space */
/* handle. */
LONG lSegid; /* Segment identifier; must */
/* be greater than 0. */
/* */
/* The name of the */
/* segment for which */
/* attribute information is to */
/* be returned. */
LONG lAttribute; /* attribute to be queried */
LONG lValue;
lAttribute = ATTR_VISIBLE;

lValue = GpiSetSegmentAttrs(hps,
                             lSegid,
                             lAttribute,
                             ATTR_ON);
```

GpiSetSegmentPriority – Set Segment Priority

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetSegmentPriority (HPS hps, LONG ISegId, LONG IRefSegId, LONG IOrder)

This function changes the position of a segment within the segment chain, or adds a segment to the chain.

Parameters

hps (HPS) – input
Presentation-space handle.

ISegId (LONG) – input
Segment identifier.

The identifier of the segment whose priority is to be changed; it must be greater than 0.

IRefSegId (LONG) – input
Reference segment identifier.

The segment that identifies a position in the segment chain. The segment specified in the *ISegId* parameter is placed either immediately before or after this segment, depending on the value specified in the *IOrder* parameter. Specifying 0 for *IRefSegId* indicates that the position is to be the beginning or the end of the segment chain as defined by the value in the *IOrder* parameter.

IOrder (LONG) – input
Segment higher or lower.

Specifies whether the segment named in the *ISegId* parameter is to be placed before or after the segment named in the *IRefSegId* parameter. Possible values are:

LOWER_PRI The segment named in the *ISegId* parameter is to have a lower priority than the segment named in the *IRefSegId* parameter. The *ISegId* segment is placed before the *IRefSegId* segment. If 0 is specified in the *IRefSegId* parameter, the segment identified in the *ISegId* parameter is placed as the highest priority segment.

HIGHER_PRI The segment named in the *ISegId* parameter is to have a higher priority than the segment named in the *IRefSegId* parameter. The *ISegId* segment is placed after the *IRefSegId* segment. If 0 is specified in the *IRefSegId* parameter, the segment identified in the *ISegId* parameter is placed as the lowest priority segment.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME An invalid segment identifier was specified.

PMERR_INV_ORDERING_PARM An invalid order parameter was specified with GpiSetSegmentPriority.

GpiSetSegmentPriority — Set Segment Priority

PMERR_SEG_AND_REFSEG_ARE_SAME	The segid and refsegid specified with GpiSetSegmentPriority were the same.
PMERR_SEG_NOT_FOUND	The specified segment identifier did not exist
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.

Remarks

The specified segment can be a segment that exists in the segment chain, or an unchained segment. The effect of this function on an unchained segment is to add it to the segment chain in the specified position.

The application may redraw the picture by drawing the segment chain (see GpiDrawChain). This causes the segments in the chain to be processed from beginning to end, so that if segments overlap, later ones are placed on top of earlier ones (assuming a default mix mode) and therefore appear to have higher priority. Changing the position of the segment in the chain therefore has the effect of changing its priority to the end user.

Related Functions

- GpiDrawChain
- GpiDrawDynamics
- GpiDrawFrom
- GpiOpenSegment
- GpiQuerySegmentPriority

Example Code

This example finds the segment with the highest priority and places a segment just before it with a lower priority.

```
#define INCL_GPISEGMENTS
#include <OS2.H>

HPS hps;          /* Presentation-space */
                  /* handle. */
LONG lRefSegid;    /* Reference-segment */
                  /* identifier. */
LONG laddSegid = 20L;
LONG lSegid;

lSegid = GpiQuerySegmentPriority(hps,
    /* find the segment with the highest */
    /* priority. */
    0,
    HIGHER_PRI);

GpiSetSegmentPriority(hps,
    lRefSegid,
    laddSegid,
    LOWER_PRI);
```

GpiSetSegmentTransformMatrix – Set Segment Transform Matrix

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
```

```
BOOL GpiSetSegmentTransformMatrix (HPS hps, LONG ISegId, LONG ICount,  
                                     PMATRIXLF pmatlfarray, LONG IOptions)
```

This function sets the segment transform that normally applies to all of the primitives in the specified segment.

Parameters

hps (**HPS**) – input
Presentation-space handle.

ISegId (**LONG**) – input
Segment identifier.

This must be greater than 0.

ICount (**LONG**) – input
Number of elements.

The number of elements to be used in the *pmatlfarray* parameter. If *ICount* is less than 9, the elements omitted default to the corresponding elements of the identity matrix (see below). Specifying *ICount* = 0 denotes that the identity matrix is used.

pmatlfarray (**PMATRIXLF**) – input
Transformation matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type **FIXED**, and have an assumed binary point between the second and third bytes. Thus, a value of 1.0 is represented by 65 536. Other elements are normal signed integers. If the presentation space coordinate format is **GPIF_SHORT** (see **GpiCreatePS**), these elements must be within the range –1 through +1.

The third, sixth, and ninth elements, when specified, must be 0, 0, and 1, respectively.

IOptions (**LONG**) – input
Transform options.

Specifies how the existing segment transform is to be modified by the transform defined by the *pmatlfarray* parameter. The new segment transform is computed, and the result stored back in the segment, replacing the existing value. When the segment is drawn, the stored segment transform is used to update the segment transform that is currently in effect, in an additive manner. Possible values are:

TRANSFORM_REPLACE The previous default segment transform is discarded and replaced by the specified transform.

TRANSFORM_ADD The specified transform is combined with the existing default segment transform, in the order (1) existing transform, (2) new transform. This option is most useful for incremental updates to transforms.

TRANSFORM_PREEMPT The specified transform is combined with the existing default segment transform, in the order (1) new transform, (2) existing transform.

GpiSetSegmentTransformMatrix —

Set Segment Transform Matrix

Returns

- Success indicator:
- TRUE** Successful completion
 - FALSE** Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_SEG_NAME	An invalid segment identifier was specified.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_SEG_NOT_FOUND	The specified segment identifier did not exist
PMERR_INV_MATRIX_ELEMENT	An invalid transformation matrix element was specified.
PMERR_INV_TRANSFORM_TYPE	An invalid options parameter was specified with a transform matrix function.

Remarks

The matrix is used to update the segment transform of a retained segment, according to the value of the *IOptions* parameter.

The segment transform is actually a model transform that applies at the start of the segment. It can be overridden later in the segment with a *GpiSetModelTransformMatrix* function.

This function specifies the transform as a one-dimensional array of *ICount* elements, being the first *ICount* elements of a 3-row by 3-column matrix ordered in rows. The order of the elements is:

Matrix	Array
<div><div><div>a</div><div>b</div><div>0</div></div><div><div>c</div><div>d</div><div>0</div></div><div><div>e</div><div>f</div><div>1</div></div></div>	(a,b,0,c,d,0,e,f,1)

The transform acts on the coordinates of the primitives in a segment, so that a point with coordinates (x,y) is transformed to the point:

$(a*x + c*y + e, b*x + d*y + f)$

The initial value of the transform of a segment is the **Identity matrix**, as shown below:

Matrix	Array
<div><div><div>1</div><div>0</div><div>0</div></div><div><div>0</div><div>1</div><div>0</div></div><div><div>0</div><div>0</div><div>1</div></div></div>	(1,0,0,0,1,0,0,0,1)

GpiSetSegmentTransformMatrix – Set Segment Transform Matrix

If scaling values greater than unity are given (which only applies if the presentation space coordinate format, as set by the GpiCreatePS function, is GPIF_LONG) it is possible for the combined effect of this and any other relevant transforms to exceed fixed-point implementation limits. This causes an error.

Segment transforms do not apply to primitives outside segments.

Related Functions

- GpiCallSegmentMatrix
- GpiQueryModelTransformMatrix
- GpiQuerySegmentTransformMatrix
- GpiSetModelTransformMatrix

Example Code

This example sets the transformation matrix of the highest priority segment to scale everything by a factor of 2.

```
#define INCL_GPISEGMENTS
#include <OS2.H>

HPS hps;          /* Presentation-space */
                  /* handle. */
LONG lSegid;      /* Segment identifier. */

MATRIXLF matlfArray = {MAKEFIXED(2,0),
                       0,0,0,MAKEFIXED(2,0),
                       0,0,0,1};

                  /* array of Transform matrix */
                  /* structures. */

lSegid = GpiQuerySegmentPriority(hps,
                                /* find the segment with the highest */
                                /* priority. */
                                0,
                                HIGHER_PRI);

GpiSetSegmentTransformMatrix(hps,
                             lSegid,
                             9L,
                             &matlfArray);
```

GpiSetStopDraw — Set Stop Draw

```
#define INCL_GPICONTROL /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetStopDraw (HPS hps, LONG IValue)

This function sets or clears the “stop draw” condition.

Parameters

hps (HPS) — input

Presentation-space handle.

IValue (LONG) — input

Stop draw condition:

SDW_OFF Clear the “stop draw” condition

SDW_ON Set the “stop draw” condition.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_INV_STOP_DRAW_VALUE

An invalid value parameter was specified with GpiSetStopDraw.

PMERR_INV_MICROPS_FUNCTION

An attempt was made to issue a function that is invalid in a micro presentation space.

Remarks

This function allows an application to set up, and control, an asynchronous thread on which long drawing operations may be performed. At the point at which the controlling thread stops a draw, it sets the “stop draw” condition. The controlling thread clears this condition after it has received an acknowledgment from the drawing thread.

The “stop draw” condition has no effect on any other calls.

If one of the following calls is made (or has already been initiated from another thread) to the same presentation space, the call is terminated if the “stop draw” condition exists:

- GpiDrawChain
- GpiDrawDynamics
- GpiDrawFrom
- GpiDrawSegment
- GpiPlayMetaFile
- GpiPutData.

The call terminates with a warning.

Any call other than GpiSetStopDraw, directed at a presentation space that is currently in use, gives a PMERR_PS_BUSY error condition.

Note: If this function is issued when an asynchronous draw to a metafile is taking place, the result is an unusable metafile.

GpiSetStopDraw — Set Stop Draw

Related Functions

- GpiDrawChain
- GpiDrawDynamics
- GpiDrawFrom
- GpiDrawSegment
- GpiPlayMetaFile
- GpiPutData
- GpiQueryStopDraw

Example Code

This example shows how to stop drawing.

```
#define INCL_GPICONTROL
#include <OS2.H>
HPS hps;          /* Presentation-space */
                  /* handle.          */

GpiSetStopDraw(hps,SDW_OFF);
```


GpiSetTag — Set Tag

```
#define INCL_GPICORRELATION /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetTag (HPS hps, LONG ITag)

This function specifies a tag by which the following primitives are to be known.

Parameters

hps (HPS) — input
Presentation-space handle.

ITag (LONG) — input
Tag identifier.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS An invalid presentation-space handle was specified.

PMERR_PS_BUSY An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION An attempt was made to issue a function that is invalid in a micro presentation space.

Remarks

When GpiCorrelateChain, GpiCorrelateFrom, or GpiCorrelateSegment is used to locate an object, both the segment identifier and the primitive tag of the object are returned to the application program.

If a tag of 0 is specified, the primitives have no name and are not returned by the correlate call.

Initially, the default and current tag are 0. The default tag can be changed with GpiSetDefTag.

Primitives within an unnamed segment cannot be picked or correlated, and any tag applied to them is ignored.

This function is not allowed between GpiBeginArea and GpiEndArea calls, therefore, all primitives within an area have the same tag.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the tag is preserved.

Related Functions

- GpiQueryDefTag
- GpiQueryTag
- GpiSetDefTag
- GpiCorrelateChain

Graphic Elements and Orders

Element Type: **OCODE_GSPIK**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Pick Identifier**

Element Type: **OCODE_GPSPK**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Pick Identifier**

Example Code

This example opens a segment and calls GpiSetTag so that all of the following primitives will be associated with that tag.

```
#define INCL_GPICORRELATION
#include <OS2.H>
HPS hps;          /* Presentation-space */
                  /* handle. */
```

```
GpiOpenSegment(hps, 0L);
GpiSetTag(hps, 0L);
```

GpiSetTextAlignment —

Set Text Alignment

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetTextAlignment (HPS hps, LONG IHorizontal, LONG IVertical)

This function determines the alignment used to position the characters in a string.

Parameters

hps (HPS) — input
Presentation-space handle.

IHorizontal (LONG) — input
Horizontal alignment:

This parameter and the next one (*IVertical*) specify the alignment of character strings horizontally and vertically. Together they define a reference point within the string that is positioned on the starting point specified for the string.

Note: The terms used in this definition (left, right, top and bottom) must be interpreted with regard to the direction of the current coordinate system, as follows:

Left	the side of the display corresponding to the lowest x-value.
Right	the side of the display corresponding to the highest x-value.
Top	the side of the display corresponding to the highest y-value.
Bottom	the side of the display corresponding to the lowest y-value.

TA_NORMAL_HORIZ Normal alignment. This is the initial default. The alignment assumed depends on the current character direction as set by GpiSetCharDirection:

CHDIRN_LEFTRIGHT	Same as TA_LEFT.
CHDIRN_TOPBOTTOM	Same as TA_US.CENTER.
CHDIRN_RIGHTLEFT	Same as TA_RIGHT.
CHDIRN_BOTTOMTOP	Same as TA_CENTER.

TA_LEFT Left alignment. The string is aligned on the left edge of its leftmost character.

TA_CENTER Center alignment. The string is aligned on the arithmetic mean of Left and Right.

TA_RIGHT Right alignment. The string is aligned on the right edge of its rightmost character.

TA_STANDARD_HORIZ Standard alignment. This is the initial default. The alignment assumed depends on the current character direction:

CHDIRN_LEFTRIGHT	Same as TA_LEFT.
CHDIRN_TOPBOTTOM	Same as TA_US.LEFT.
CHDIRN_RIGHTLEFT	Same as TA_RIGHT.
CHDIRN_BOTTOMTOP	Same as TA_LEFT.

GpiSetTextAlignment – Set Text Alignment

IVertical (LONG) – input
Vertical alignment:

TA_NORMAL_VERT

Normal alignment. This is the initial default. The alignment assumed depends on the current character direction as set by `GpiSetCharDirection`:

CHDIRN_LEFTRIGHT	Same as TA_BASE .
CHDIRN_TOPBOTTOM	Same as TA_US.TOP .
CHDIRN_RIGHTLEFT	Same as TA_BASE .
CHDIRN_BOTTOMTOP	Same as TA_BOTTOM .

TA_TOP

Top alignment. The string is aligned on the top edge of its topmost character.

TA_HALF

Half alignment. The string is aligned on the arithmetic mean of Bottom and Top.

TA_BASE

Base alignment. The string is aligned on the base of its bottom character.

TA_BOTTOM

Bottom alignment. The string is aligned on the bottom edge of its bottom character.

TA_STANDARD_VERT

Standard alignment. This is the initial default. The alignment assumed depends on the current character direction:

CHDIRN_LEFTRIGHT	Same as TA_BOTTOM .
CHDIRN_TOPBOTTOM	Same as TA_US.TOP .
CHDIRN_RIGHTLEFT	Same as TA_BOTTOM .
CHDIRN_BOTTOMTOP	Same as TA_BOTTOM .

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from `WinGetLastError`

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_CHAR_ALIGN_ATTR

The text alignment attribute specified in `GpiSetTextAlignment` is not valid.

Remarks

This function must not be issued in an area bracket. The attribute mode determines whether the current value of the text alignment attribute is preserved.

Support for this function is device dependent.

GpiSetTextAlignment —

Set Text Alignment

Related Functions

- GpiQueryTextAlignment
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix
- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt

Graphic Elements and Orders

Element Type: **OCODE_GSTA**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Text Alignment**

Element Type: **OCODE_GPSTA**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Text Alignment**

GpiSetViewingLimits – Set Viewing Limits

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetViewingLimits (HPS hps, PRECTL prclLimits)

This function establishes a clipping rectangle in model space.

Parameters

hps (HPS) – input
Presentation-space handle.

prclLimits (PRECTL) – input
Viewing limits in model space.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.

Remarks

Viewing limits can be set within a segment, and apply to all subsequent primitives in the segment and any segments it calls. They can be changed at any time within the segment and they are not subject to segment or model transformations. Limits specified in called segments override those set by the limits of the root segment.

The limits are reset to their default value at the start of each root segment, subject to the fast-chaining attribute, like primitive attributes. The initial default value is no clipping; this can be changed with GpiSetDefViewingLimits.

The boundaries are inclusive, so that points on them are not clipped (removed). If either the left boundary of *prclLimits* is greater than the right, or the bottom greater than the top, a NULL rectangle is defined. All points are clipped.

Attribute mode (see GpiSetAttrMode) has no effect on this function.

The viewing limits are converted under the current viewing and default viewing transformations to a clipping rectangle in the page. This remains in force until changed by a subsequent GpiSetViewingLimits function. Clipping actually takes place to the intersection of the viewing limits, the clip path, the clip region, the graphics field, and the client area on the device.

GpiSetViewingLimits — Set Viewing Limits

Related Functions

- GpiQueryViewingLimits
- GpiSetDefViewingLimits
- GpiSetGraphicsField

Graphic Elements and Orders

Element Type: **OCODE_GSVW**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: **Set Viewing Window**

Element Type: **OCODE_GPSVW**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: **Push and Set Viewing Window**

Example Code

In this example the model space clipping region width is reduced to 400x400.

```
#define INCL_GPITRANSFORMS
#include <OS2.H>

HPS hps;          /* Presentation-space */
                  /* handle.          */

BOOL fSuccess;
RECTL rcLimits = { /* viewing limits. */
                  10,10,
                  410,410
                  };

fSuccess = GpiSetViewingLimits(hps,
                               &rcLimits);
```

GpiSetViewingTransformMatrix – Set Viewing Transform Matrix

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiSetViewingTransformMatrix (HPS hps, LONG ICount, PMATRIXLF pmatlfArray, LONG IOptions)

This function sets the viewing transform that is to apply to any subsequently opened segments.

Parameters

hps (HPS) – input
Presentation-space handle.

ICount (LONG) – input
Number of elements.

The number of elements supplied in *pmatlfArray*, that are to be examined, starting from the beginning of the structure. If *ICount* is less than 9, remaining elements default to the corresponding elements of the identity matrix. Specifying *ICount* = 0 means that the identity matrix is used.

pmatlfArray (PMATRIXLF) – input
Transformation matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type FIXED, and have an assumed binary point between the second and third bytes. Thus a value of 1.0 is represented by 65 536. Other elements are normal signed integers. If the presentation space coordinate format is GPIF_SHORT (see GpiCreatePS), these elements must be within the range -1 through +1.

The third, sixth, and ninth elements, when specified, must be 0, 0, and 1, respectively.

IOptions (LONG) – input
Transform option.

Specifies how the specified transform is to be used to modify the existing viewing transform. This must be:

TRANSFORM_REPLACE New and replace. The previous viewing transform is discarded and replaced by the specified transform.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION

An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_LENGTH_OR_COUNT

An invalid length or count parameter was specified.

PMERR_INV_MATRIX_ELEMENT

An invalid transformation matrix element was specified.

PMERR_INV_TRANSFORM_TYPE

An invalid options parameter was specified with a transform matrix function.

GpiSetViewingTransformMatrix — Set Viewing Transform Matrix

PMERR_INV_IN_SEG

An attempt was made to issue a function invalid inside a segment bracket.

PMERR_NOT_IN_RETAIN_MODE

An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to **retain**

Remarks

This function is only valid outside segments. The viewing transform that is set applies to all subsequently opened (new) segments (it has no effect on primitives outside segments). All graphics primitives in a segment must have the same viewing transform. When it has been set for a particular segment, the viewing transform for that segment cannot be changed.

The transform is specified as a one-dimensional array of *ICount* elements, being the first *n* elements of a 3-row by 3-column matrix ordered by rows. The order of the elements is:

Matrix

Array

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix}$$

(a,b,0,c,d,0,e,f,1)

The transform acts on the coordinates of the primitives in a segment, so that a point with coordinates (x,y) is transformed to the point:

(a*x + c*y + e, b*x + d*y + f)

The initial value of the viewing transform is the identity matrix, as shown below:

Matrix

Array

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(1,0,0,0,1,0,0,0,1)

The viewing transform must be set (or defaulted) to the unity transform, before any segment that is to be called is first opened.

If scaling values greater than unity are given (which only applies if the presentation space coordinate format, as set by the GpiCreatePS function, is GPIF_LONG) it is possible for the combined effect of this and any other relevant transforms to exceed fixed-point implementation limits. This causes an error.

This function must not be issued in a path or area bracket.

Related Functions

- GpiQueryViewingTransformMatrix
- GpiSetDefaultViewMatrix

GpiSetViewingTransformMatrix — Set Viewing Transform Matrix

Example Code

In this example, the GpiSetViewingTransformMatrix is used to replace the existing viewing transformation. The new transformation will then double the width and height of drawing.

```
#define INCL_GPITRANSFORMS
#include <OS2.H>

HPS hps;          /* Presentation space handle. */
LONG lCount;      /* maximum number of elements */
MATRIXLF matlf = { MAKEFIXED(2,0), /* scale x coordinates by a */
                  /* factor of 2. */
                  0, 0, 0, /* no rotation. */
                  MAKEFIXED(2,0), /* scale y coordinates by a */
                  /* factor of 2. */
                  0, 0, 0, 1}; /* no rotation. */

GpiSetViewingTransformMatrix(hps,
                             9L, /* number of elements. */
                             &matlf,
                             TRANSFORM_REPLACE);
```

GpiStrokePath – Stroke Path

```
#define INCL_GPIPATHS /* Or use INCL_GPI or INCL_PM */
```

LONG GpiStrokePath (HPS hps, LONG lPath, ULONG flOptions)

This function strokes a path, and then draws it.

Parameters

hps (HPS) – input

Presentation-space handle.

lPath (LONG) – input

Identifier of path to be stroked; it must be 1.

flOptions (ULONG) – input

Stroke option:

Reserved; must be 0.

Returns

Correlation and error indicators:

GPI_OK Successful

GPI_HITS Correlate hits

GPI_ERROR Error.

Possible returns from WinGetLastError

PMERR_INV_HPS

An invalid presentation-space handle was specified.

PMERR_PS_BUSY

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PATH_ID

An invalid path identifier parameter was specified.

PMERR_INV_RESERVED_FIELD

An invalid reserved field was specified.

PMERR_PATH_UNKNOWN

An attempt was made to perform a path function on a path that did not exist.

Remarks

The path is first converted to one that describes the envelope of a wide line stroked using the current geometric line-width attribute (see GpiSetLineWidthGeom).

Note: This function and GpiModifyPath are the only calls that can cause geometric wide lines to be constructed. For more details about the way in which the envelope is constructed, see GpiModifyPath.

The converted path is then filled, using winding mode area fill and the area attributes. The boundaries of the wide line are included in the fill.

When it has been drawn, the path is deleted.

This function is equivalent to GpiModifyPath, followed by GpiFillPath. It is provided to enable device drivers to optimize storage, if possible.

If the current drawing mode (see GpiSetDrawingMode) is **draw** or **draw-and-retain**, drawing occurs on the currently associated device. If the drawing mode is **retain**, this function is stored in the current segment and output occurs when the segment is subsequently drawn in the usual way.

Related Functions

- GpiBeginArea
- GpiBeginPath
- GpiEndPath
- GpiFillPath
- GpiModifyPath
- GpiOutlinePath
- GpiPathToRegion
- GpiSetClipPath
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom

Graphic Elements and Orders

Element Type: **OCODE_GFPTH**

Note that GpiFillPath also generates this element type.

Order: **Fill Path**

Example Code

This example uses the GpiStrokePath function to draw a wide line.

```
#define INCL_GPIPATHS
#include <OS2.H>
HPS hps;          /* Presentation space handle. */
POINTL ptlStart = { 0, 0 };
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 };

/* create the path */

GpiBeginPath(hps, 1L);
GpiMove(hps, &ptlStart);
GpiPolyLine(hps, 3, ptlTriangle);
GpiEndPath(hps);

GpiSetLineWidthGeom(hps, 20L); /* set the line width */
GpiStrokePath(hps, 1L, 0L);    /* draw the wide line */
```

GpiTranslate – Translate Matrix

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI or INCL_PM */
```

**BOOL GpiTranslate (HPS hps, PMATRIXLF pmatlfArray, LONG IOptions,
PPOINTL pptlTranslation)**

This function applies a translation to a transform matrix.

Parameters

hps (HPS) – input
Presentation-space handle.

pmatlfArray (PMATRIXLF) – input/output
Transform matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type FIXED, and have an assumed binary point between the second and third bytes. Thus a value of 1.0 is represented by 65 536. Other elements are normal signed integers.

The third, sixth, and ninth elements must be 0, 0, and 1, respectively.

IOptions (LONG) – input
Transform options.

Specifies how the transform defined by the specified translation should be used to modify the previous transform specified by the *pmatlfArray* parameter. Possible values are:

TRANSFORM_REPLACE The previous transform is discarded and replaced by the transform describing the specified translation.

TRANSFORM_ADD The previous transform is combined with a transform representing the specified translation in the order (1) previous transform, (2) translational transform. This option is most useful for incremental updates to transforms.

pptlTranslation (PPOINTL) – input
Translation.

The coordinates of a point, relative to the origin, which defines the required translation.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INV_TRANSFORM_TYPE An invalid options parameter was specified with a transform matrix function.

Remarks

This function is a helper function which either applies a specified translational component to an existing transform matrix, or replaces the matrix with one that represents the specified translation alone.

The transform is specified as a one-dimensional array of 9 elements that are the elements of a 3-row by 3-column matrix ordered by rows. The order of the elements are as follows:

Matrix

Array

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix}$$

(a,b,0,c,d,0,e,f,1)

Transforms act on the coordinates of primitives, so that a point with coordinates (x,y) is transformed to the point:

$$(a*x + c*y + e, b*x + d*y + f)$$

The transform can be used in any call following:

- GpiSetModelTransformMatrix
- GpiSetSegmentTransformMatrix
- GpiSetViewingTransformMatrix
- GpiSetDefaultViewMatrix.

Other similar helper functions are:

- GpiScale to apply a scaling component
- GpiRotate to apply a rotation component.

Related Functions

- GpiRotate
- GpiScale
- GpiSetModelTransformMatrix
- GpiSetSegmentTransformMatrix
- GpiSetDefaultViewMatrix
- GpiSetViewingTransformMatrix

GpiTranslate — Translate Matrix

Example Code

This example translates the center of the picture back to the center of the page.

```
#define INCL_GPITRANSFORMS
#define INCL_WINSYS
#include <OS2.H>

HPS hps;          /* Presentation space handle. */
MATRIXLF matlf;   /* Current viewing transformation */
POINTL ptlPictCenter;

/* determine the center of the page */
ptlPictCenter.x = WinQuerySysValue(HWND_DESKTOP,
                                   SV_CXFULLSCREEN)/2 - ptlPictCenter.x;
ptlPictCenter.y = WinQuerySysValue(HWND_DESKTOP,
                                   SV_CYFULLSCREEN)/2 - ptlPictCenter.y;

GpiQueryViewingTransformMatrix(hps,
                               9L,
/* Translate the center of the picture back to the center of the */
/* page.                                                         */
                               &matlf);

GpiTranslate(hps,
             &matlf,
             TRANSFORM_ADD,
             &ptlPictCenter);
```

```
#define INCL_GPILCIDS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiUnloadFonts (HAB hab, PSZ pszFilename)

This function unloads any fonts previously loaded from the resource file by GpiLoadFonts.

Parameters

hab (HAB) – input
Anchor-block handle.

pszFilename (PSZ) – input
Fully qualified file name of the font resource.
The file name extension is .FON

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_FONT_FILE_NOT_LOADED	An attempt was made to unload a font file that was not loaded.
PMERR_OWN_SET_ID_REFS	An attempt to unload a font failed because the setid is still being referenced.

Remarks

Before issuing this function, the application must:

1. Issue GpiSetCharSet to a font other than one of those to be unloaded, for example, to the default font.
2. Issue GpiDeleteSetId for each local identifier (lcid) that references one of the fonts (the LCID_ALL option can be used if all lcid's are to be deleted).

An error is returned if lcid's that reference one of the fonts still exist for this application, and a warning is logged if lcid's exist for another application.

Related Functions

Prerequisite Functions

- GpiDeleteSetId
- GpiSetCharSet

Other Related Functions

- GpiCreateLogFont
- GpiLoadFonts
- GpiQueryFontMetrics
- GpiQueryFonts
- GpiQueryKerningPairs
- GpiQueryNumberSetIds
- GpiQuerySetIds
- GpiQueryWidthTable

GpiUnloadFonts — Unload Fonts

Example Code

This function unloads any font(s) previously loaded from the resource file by GpiLoadFonts.

```
#define INCL_GPILCIDS  
#include <OS2.H>
```

```
HAB hab;          /* Anchor-block handle. */  
char fntname[] = "HELVETICA.FON";
```

```
GpiUnloadFonts(hab, fntname);
```

GpiUnloadPublicFonts – Unload Public Fonts

```
#define INCL_GPILCIDS /* Or use INCL_GPI or INCL_PM */
```

BOOL GpiUnloadPublicFonts (HAB hab, PSZ pszFilename)

This function unloads one or more generally-available fonts from the specified resource file. See GpiLoadPublicFonts.

Parameters

hab (HAB) – input
Anchor-block handle.

pszFilename (PSZ) – input
Filename.

This is the fully-qualified name of the font resource. The file-name extension is .FON

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_FONT_FILE_NOT_LOADED

An attempt was made to unload a font file that was not loaded.

PMERR_SET_ID_REFS

An attempt to unload a font failed because the setid is still being referenced.

Remarks

Before issuing this function, the application must:

1. Issue GpiSetCharSet to a font other than one of those to be unloaded, for example, to the default font.
2. Issue GpiDeleteSetId for each local identifier (lcid) that references one of the fonts (the LCID_ALL option can be used if all lcid's are to be deleted).

An error is returned if lcid's that reference one of the fonts still exist for this or any other application, and the unload fails.

Note: If another application is using the fonts when this function is issued, so that the call fails, the fonts are likely to remain loaded until the next boot. This is true even if the other application has issued a GpiLoadPublicFonts, and will later issue a GpiUnloadPublicFonts, since the use count is not decremented when the call fails.

It is also possible for one application to get details of the fonts with GpiQueryFonts, but then to fail to be able to use them with GpiCreateLogFont, because another application unloaded them with GpiUnloadPublicFonts in the time between the two calls.

GpiUnloadPublicFonts — Unload Public Fonts

Example Code

This function unloads one or more generally-available fonts from the specified resource file.

```
#define INCL_GPILCIDS  
#include <OS2.H>
```

```
HAB hab;          /* Anchor-block handle. */  
char fntname[] = "HELVETICA.FON";
```

```
GpiUnloadPublicFonts(hab, fntname);
```

GpiWCBitBlt – World Coordinates Bit Blt

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI or INCL_PM. Also in COMMON section */
```

```
LONG GpiWCBitBlt (HPS hpsTarget, HBITMAP hbmSource, LONG ICount, PPOINTL aptlPoints,  
LONG IROP, ULONG flOptions)
```

This function copies a rectangle of bit-map image data.

Parameters

hpsTarget (HPS) – input
Target presentation-space handle.

hbmSource (HBITMAP) – input
Source bit-map handle.

It is an error if this bit map is currently selected into a memory device context.

ICount (LONG) – input
Point count.
This count must be equal to 4.

aptlPoints (PPOINTL) – input
Point array

Array of *ICount* points, in the order **Tx1, Ty1, Tx2, Ty2, Sx1, Sy1, Sx2, Sy2**. These are:

Tx1,Ty1 Specify the bottom-left corner of the target rectangle in target world coordinates.

Tx2,Ty2 Specify the top-right corner of the target rectangle in target world coordinates.

Sx1,Sy1 Specify the bottom-left corner of the source rectangle in source device coordinates.

Sx2,Sy2 Specify the top-right corner of the source rectangle in source device coordinates.

IROP (LONG) – input
Mixing function required.

Each plane of the target can be considered to be processed separately. For any pel in a target plane, three bits together with the *IROP* values are used to determine the final value. These are the value of that pel in the Pattern (P) and Source (S) data and the initial value of that pel in the Target (T) data. For any combination of P, S, and T pel values, the final target value for the pel is determined by the appropriate *IROP* bit value as shown below:

P	S	T (Initial)	T (final)
0	0	0	Index bit 0 (least significant)
0	0	1	Index bit 1
0	1	0	Index bit 2
0	1	1	Index bit 3
1	0	0	Index bit 4
1	0	1	Index bit 5
1	1	0	Index bit 6
1	1	1	Index bit 7 (most significant)

GpiWCBitBlt — World Coordinates Bit Blt

The index formed in the above way determines the mixing required. Mnemonic names are available for commonly used mixes:

ROP_SRCOPY	/* SRC	*/
ROP_SRCPAINT	/* SRC OR DST	*/
ROP_SRCAND	/* SRC AND DST	*/
ROP_SRCINVERT	/* SRC XOR DST	*/
ROP_SRCERASE	/* SRC AND NOT(DST)	*/
ROP_NOTSRCOPY	/* NOT(SRC)	*/
ROP_NOTSRCERASE	/* NOT(SRC) AND NOT(DST)	*/
ROP_MERGECOPY	/* SRC AND PAT	*/
ROP_MERGEPAINT	/* NOT(SRC) OR DST	*/
ROP_PATCOPY	/* PAT	*/
ROP_PATPAINT	/* NOT(SRC) OR PAT OR DST	*/
ROP_PATINVERT	/* DST XOR PAT	*/
ROP_DSTINVERT	/* NOT(DST)	*/
ROP_ZERO	/* 0	*/
ROP_ONE	/* 1	*/

fOptions (ULONG) — input
Options.

How eliminated lines or columns are treated if a compression is performed.

Flags 15 through 31 of *fOptions* can be used for privately-supported modes for particular devices.

BBO_OR The default. If compression is necessary, logical-OR eliminated rows or columns. This is useful for white on black.

BBO_AND If compression is necessary, logical-AND eliminated rows or columns. This is useful for black on white.

BBO_IGNORE If compression is necessary, ignore eliminated rows or columns. This is useful for color.

Returns

Correlation and error indicators:

GPI_OK	Successful
GPI_HITS	Correlate hits
GPI_ERROR	Error.

Possible returns from WinGetLastError

PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_INV_BITBLT_MIX	An invalid <i>IRop</i> parameter was specified with a GpiBitBlt or GpiWCBitBlt function.
PMERR_INV_BITBLT_STYLE	An invalid options parameter was specified with a GpiBitBlt or GpiWCBitBlt function.
PMERR_BITMAP_NOT_FOUND	A attempt was made to perform a bit-map operation on a bit map that did not exist.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.
PMERR_INV_RECT	An invalid rectangle parameter was specified.
PMERR_NO_BITMAP_SELECTED	An attempt has been made to operate on a memory device context that has no bit map selected.

GpiWCBitBlt — World Coordinates Bit Blt

PMERR_INCORRECT_DC_TYPE	An attempt was made to perform a bit-map operation on a presentation space associated with a device context of a type that is unable to support bit-map operations.
PMERR_INCOMPATIBLE_BITMAP	An attempt was made to select a bit map or perform a BitBlt operation on a device context that was incompatible with the format of the bit map.
PMERR_INV_HBITMAP	An invalid bit-map handle was specified.
PMERR_HBITMAP_BUSY	An internal bit map busy error was detected. The bit map was locked by one thread during an attempt to access it from another thread.

Remarks

A rectangle of bit-map image data is copied from a bit map, to a bit map selected into a device context associated with the target presentation space. Alternatively, the target presentation space can be associated with a device context that specifies a suitable raster device, for example, the screen.

Note: In either case, both source and target device contexts must apply to the same physical device. It is an error if this device does not support raster operations.

A rectangle is specified in device coordinates for the source bit map, and one in world coordinates for the target presentation space. The source rectangle is noninclusive; the left and lower boundaries in device space are included, but not the right and upper boundaries. Thus if the bottom-left is equal to the top-right, the rectangle is deemed to be empty. The target rectangle is "inclusive-inclusive"; that is, all boundaries are included in the rectangle.

If the target rectangle, after transformation to device coordinates and adjustment for inclusivity, is not the same size as the source rectangle, then stretching or compressing of the data occurs. *fOptions* specifies how eliminated rows or columns of bits are to be treated if compression occurs. Note that the pattern data is never stretched or compressed.

If there is a rotational effect in the transforms, the copy of the bit map is rotated accordingly.

The target rectangle is transformed to device coordinates, and if any shear or rotation has occurred, this is then converted to an upright rectangle that bounds the transformed figure. This rectangle is used as the target of the operation. No inversion of the image takes place.

These current attributes of the target presentation space are used (other than for converting between monochrome and color, as described below):

- Area color
- Area background color
- Pattern set
- Pattern symbol.

The color values are used in conversion between monochrome and color data. This is the only format conversion performed by this function. The conversions are:

- Output of a monochrome pattern to a color device.

In this instance the pattern is converted first to a color pattern, using the current area colors:

- source 1s → area foreground color
- source 0s → area background color.

- Copying from a monochrome bit map to a color bit map (or device).

The source bits are converted as follows:

- source 1s → image foreground color
- source 0s → image background color.

GpiWCBitBlt — World Coordinates Bit Blt

- Copying from a color bit map to a monochrome bit map (or device).

The source bits are converted as follows:

- source nonzeros → image foreground color
- source 0s → image background color.

If the mix (*/Rop*) does not call for a pattern, the pattern set and pattern symbol are not used.

Neither the source nor the pattern is required when a bit map, or part of a bit map, is to be cleared to a particular color.

If the mix does require both source and pattern, a three-way operation is performed.

If a pattern is required, dithering may be performed for solid patterns in a color that is not available on the device. See `GpiSetPattern`.

This function (unlike `GpiBitBlt`) can be drawn immediately, retained in segment store, or both of these, depending upon the drawing mode (see `GpiSetDrawingMode`).

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see “Metafile Restrictions” on page G-1.

Related Functions

- `DevQueryCaps`
- `GpiBitBlt`
- `GpiCreateBitmap`
- `GpiDeleteBitmap`
- `GpiDrawBits`
- `GpiLoadBitmap`
- `GpiQueryBitmapBits`
- `GpiQueryBitmapDimension`
- `GpiQueryBitmapHandle`
- `GpiQueryBitmapParameters`
- `GpiQueryDeviceBitmapFormats`
- `GpiSetBitmap`
- `GpiSetBitmapBits`
- `GpiSetBitmapDimension`
- `GpiSetBitmapId`
- `WinDrawBitmap`
- `WinGetSysBitmap`

Graphic Elements and Orders

Element Type: `OCODE_GBBLT`

Order: `Bitblt`

Example Code

This function copies a rectangle of bit-map image data. This example uses GpiWCBitBlt to copy and compress a bit map in a presentation space. The function copies the bit map that is 100 pels wide and 100 pels high into a 50-by-50-pel rectangle at the location (300,400). Since the raster operation is ROP_SRCCOPY, GpiWCBitBlt replaces the image previously in the presentation-space rectangle. The function compresses the bit map to fit the new rectangle by discarding extra rows and columns as specified by the BBO_IGNORE option.

```
#define INCL_GPIBITMAPS
#include <OS2.H>
HPS hps;          /* Presentation space handle. */
HBITMAP hbm;
POINTL aptl[4] = {
    300, 400,      /* lower-left corner of target      */
    350, 450,      /* upper-right corner of target     */
    0, 0,          /* lower-left corner of source     */
    100, 100 };    /* upper-right corner of source     */

GpiWCBitBlt(hps, /* presentation space */
            hbm,  /* bit-map handle */
            4L,   /* four points needed to compress */
            aptl, /* points for source and target rectangles */
            ROP_SRCCOPY, /* copy source replacing target */
            BBO_IGNORE); /* discard extra rows and columns */
```

Chapter 6. Profile Functions

The following table shows all the Profile (Prf) functions in alphabetic order.

C Name
PrfCloseProfile
PrfOpenProfile
PrfQueryProfile
PrfQueryProfileData
PrfQueryProfileInt
PrfQueryProfileSize
PrfQueryProfileString
PrfReset
PrfWriteProfileData
PrfWriteProfileString

PrfCloseProfile — Close Profile

```
#define INCL_WINSHELLDATA /* Or use INCL_WIN or INCL_PM */
```

BOOL PrfCloseProfile (HINI hIni)

This function indicates that a profile is no longer available for use.

Parameters

hIni (HINI) – input
Initialization-file handle.

After this function, the handle is no longer valid.

Returns

Success indicator:

TRUE Successful completion.

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INI_FILE_IS_SYS_OR_USER User or system initialization file cannot be closed.

PMERR_INVALID_INI_FILE_HANDLE An invalid initialization-file handle was specified.

Remarks

This function cannot be used to close the current user or system initialization files.

Related Functions

- PrfOpenProfile

Example Code

This example calls PrfCloseProfile to close a profile and makes it unavailable for use.

```
#define INCL_WINSHELLDATA      /* Window Shell functions      */
#include <os2.h>

BOOL fSuccess;                /* success indicator      */
HINI hini;                    /* initialization-file handle */

fSuccess = PrfCloseProfile(hini);
```

PrfOpenProfile – Open Profile

```
#define INCL_WINSHELLDATA /* Or use INCL_WIN or INCL_PM */
```

HINI PrfOpenProfile (HAB hab, PSZ pszFileName)

This function indicates that a file is available for use as a profile.

Parameters

hab (HAB) – input
Anchor-block handle.

pszFileName (PSZ) – input
User-profile file name.

This must not be the same as the current user or system initialization file name.

Returns

Initialization-file handle.

This handle is used on other calls to manipulate the profile file.

NULLHANDLE Error occurred

Other Initialization-file handle.

Possible returns from WinGetLastError

PMERR_OPENING_INI_FILE Unable to open initialization file (due to lack of disk space for example).

PMERR_MEMORY_ALLOC An error occurred during memory management.

PMERR_INI_FILE_IS_SYS_OR_USER User or system initialization file cannot be closed.

Remarks

A user profile and a system profile are opened by the system, either at start-up time, or (in the case of the user profile) as a result of a PrfReset function, and are always available. Their handles are HINI_USERPROFILE and HINI_SYSTEMPROFILE. Applications do not have to open or close the user profile or the system profile.

The handle returned is only valid for the process issuing the PrfOpenProfile function.

The PrfOpenProfile function can be used by an administrator's application that is creating or modifying a profile for a user.

It can also be used to create a back-up profile as follows:

- Use the enumerate form of PrfQueryProfileData to obtain a list of application names in the profile being backed up.
- Use the enumerate form of PrfQueryProfileData to obtain a list of key names for each of the application names.
- Use PrfQueryProfileData for each application-name or key-name pair to read the appropriate data.
- Use PrfWriteProfileData to write the data into the back-up profile.

PrfOpenProfile — Open Profile

Related Functions

- PrfCloseProfile
- PrfQueryProfileData

Example Code

This example uses PrfOpenProfile to open and make available a profile for the file 'PROFILE.INI'.

```
#define INCL_WINHELLDATA      /* Window Shell functions      */
#include <os2.h>

HINI hini;                    /* initialization-file handle      */
HAB  hab;                     /* anchor-block handle            */
char pszFileName[13]; /* user-profile file name      */

strcpy(pszFileName,"PROFILE.INI");

hini = PrfOpenProfile(hab,pszFileName);
```

```
#define INCL_WINSHELLDATA /* Or use INCL_WIN or INCL_PM */
```

BOOL PrfQueryProfile (HAB hab, PPRFPROFILE pprfproProfile)

This function returns a description of the current user and system profiles.

Parameters

hab (HAB) – input
Anchor-block handle.

pprfproProfile (PRFPROFILE) – input/output
Profile names structure.

The *cchUserName* and the *cchSysName* parameters of the PRFPROFILE data structure are set to the lengths of the respective file names, even if truncation occurs. If these fields are initialized to 0 by the application, then the *pszUserName* and *pszSysName* parameters are not inspected, and the application can then determine the sizes of the buffers required to hold the names on a second call. Otherwise, the *pszUserName* and *pszSysName* parameters must point to reserved areas of memory, and the *cchUserName* and *cchSysName* parameters must indicate the sizes of those areas.

If the *pszUserName* or the *pszSysName* parameter is NULL, then there is no defined user or system profile, respectively.

Returns

Success indicator:

TRUE Successful completion.

FALSE Error occurred, or there was insufficient space to record the names, which have been truncated.

Related Functions

- PrfReset

PrfQueryProfile — Query Profile

Example Code

This example calls PrfQueryProfile to obtain a description of the current user and system profiles, in this case querying the lengths of the user and system profile file names and placing the values in variables.

```
#define INCL_WINSHELLDATA      /* Window Shell functions      */
#include <os2.h>

BOOL fSuccess;                /* success indicator      */
HAB hab;                      /* anchor-block handle    */
PRFPROFILE pprfproProfile; /* Profile names structure */
ULONG ulUserNameLen;         /* length of user file name */
ULONG ulSysNameLen;         /* length of system file name */

/* initialize lengths so that query will return the buffer sizes*/
pprfproProfile.cchUserName = 0L;
pprfproProfile.cchSysName = 0L;

fSuccess = PrfQueryProfile(hab, &pprfproProfile);

if (fSuccess == TRUE)
{
    ulUserNameLen = pprfproProfile.cchUserName;
    ulSysNameLen = pprfproProfile.cchSysName;
}
```

PrfQueryProfileData – Query Profile Data

```
#define INCL_WINSHELLDATA /* Or use INCL_WIN or INCL_PM */
```

```
BOOL PrfQueryProfileData (HINI hIni, PSZ pszApp, PSZ pszKey, PVOID pBuffer,  
                          PULONG pulBufferMax)
```

This function returns a string of binary data from the specified profile.

Parameters

hIni (HINI) – input
Initialization-file handle.

HINI_PROFILE Both the user profile and system profile are searched

HINI_USERPROFILE The user profile is searched

HINI_SYSTEMPROFILE The system profile is searched

Other Initialization-file handle.

pszApp (PSZ) – input
Application name.

The name of the application for which the profile data is required. The name must match exactly with the name stored in the profile. There is no case-independent searching.

If this parameter is NULL, this function enumerates all the application names present in the profile and returns the names as a list in the *pBuffer* parameter. Each application name is terminated with a NULL character and the last name is terminated with two successive NULL characters. In this case, the *pulBufferMax* parameter contains the total length of the list excluding the final NULL character.

pszKey (PSZ) – input
Key name.

The name of the key for which the profile data is required. The name must match exactly with the name stored in the profile. There is no case-independent searching.

If this parameter is NULL, and if *pszApp* is not equal to NULL, this call enumerates all key names associated with the named application and returns the key names, but not their values, as a list in the *pBuffer* parameter. Each key name is terminated with a NULL character and the last name is terminated with two successive NULL characters. In this case, the *pulBufferMax* parameter contains the total length of the list **excluding** the final NULL character.

pBuffer (PVOID) – output
Value data.

A buffer in which the value corresponding to the key name is returned. The returned data is not null terminated, unless the value data is explicitly null terminated within the file. This function handles binary data.

pulBufferMax (PULONG) – input/output
Size of value data.

This is the size of the buffer specified by the *pBuffer* parameter. If the call is successful, this is overwritten with the number of bytes copied into the buffer.

PrfQueryProfileData — Query Profile Data

Returns

Success indicator:

TRUE Successful completion
FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INVALID_PARM	A parameter to the function contained invalid data.
PMERR_NOT_IN_IDX	The application name, key-name or program handle was not found.
PMERR_CAN_NOT_CALL_SPOOLER	An error occurred attempting to call the spooler validation routine. This error is not raised if the spooler is not installed.

Remarks

This function returns a string of binary data from the profile. The call searches the file for a key matching the name specified by the *pszKey* parameter, under the application heading specified by the *pszApp* parameter.

Enumeration can be performed in exactly the same way as in the *PrfQueryProfileString* function. The enumeration returns application or key names irrespective of whether the data concerned is written with the *PrfWriteProfileString* function or the *PrfWriteProfileData* function.

This function returns data that is written to the file using either the *PrfWriteProfileString* function or the *PrfWriteProfileData* function.

If the *pszApp* parameter is NULL, this call enumerates all application names and constructs in the *pBuffer* parameter a list of application names. Each application name in the list is terminated with a null character. The last string in the list is terminated with two null characters. This function returns the length of the list, up to, but not including, the final null. If the enumerated application names exceed the available buffer space, the enumerated names are truncated, the enumerated list is not terminated with 2 bytes of zeros, and the *fSuccess* parameter is set to FALSE. In this case, *pszKey* is ignored.

If the *pszApp* parameter is valid and if the *pszKey* is NULL, this function enumerates all key names associated with the *pszApp* parameter by constructing in the *pBuffer* parameter a list of key names. Each key name in the list is terminated with a null character. The last string in the list is terminated with two null characters. This function returns the length of the list, up to, but not including, the final null. If the enumerated key names exceed the available buffer space, the enumerated names are truncated, the enumerated list is not terminated with 2 bytes of zeros, and the *fSuccess* parameter is set to FALSE.

Related Functions

- *PrfQueryProfileSize*
- *PrfWriteProfileData*

PrfQueryProfileData – Query Profile Data

Example Code

This example calls PrfQueryProfileData to search the user and system profiles for the value of key 'KEY' within the application 'APP' and return the value if found.

```
#define INCL_WINSHELLDATA      /* Window Shell functions      */
#include <os2.h>

BOOL fSuccess;                /* success indicator      */
HINI hini;                    /* initialization-file handle */
char pszApp[10];              /* application name        */
char pszKey[10];              /* key name                */
VOID *pBuffer;               /* Value data              */
ULONG pulBufferMax;          /* Size of value data      */

/* Both the user profile and system profile are searched */
hini = HINI_PROFILE;

/* specify application and key names */
strcpy(pszApp, "APP");
strcpy(pszKey, "KEY");

fSuccess = PrfQueryProfileData(hini, pszApp, pszKey, pBuffer,
                               &pulBufferMax);
```

PrfQueryProfileInt – Query Profile Integer

```
#define INCL_WINSHELLDATA /* Or use INCL_WIN or INCL_PM */
```

LONG PrfQueryProfileInt (HINI hIni, PSZ pszApp, PSZ pszKey, LONG IDefault)

This function returns an integer value from the specified profile.

Parameters

hIni (HINI) – input

Initialization-file handle.

HINI_PROFILE

Both the user profile and system profile are searched

HINI_USERPROFILE

The user profile is searched

HINI_SYSTEMPROFILE

The system profile is searched

Other

Initialization-file handle.

pszApp (PSZ) – input

Application name.

The name of the application for which the profile data is required. The name must match exactly with the name stored in the profile. There is no case-independent searching.

pszKey (PSZ) – input

Key name.

The name of the key for which the profile data is required. The name must match exactly with the name stored in the profile. There is no case-independent searching.

IDefault (LONG) – input

Default value.

This value is returned in *IResult*, if the key defined by *pszKey* cannot be found in the initialization file.

Returns

Key value specified in the initialization file.

The value of the key specified by *pszKey* in the initialization file.

If the value corresponding to the key is not an integer, *IResult* is 0.

If the key-name value is a series of digits followed by non-numeric characters, *IResult* contains the value of the digits only. For example, "KeyName=102abc" causes the value 102 to appear in *IResult*.

Possible returns from WinGetLastError

PMERR_INVALID_PARAM

A parameter to the function contained invalid data.

PMERR_NOT_IN_IDX

The application name, key-name or program handle was not found.

PMERR_CAN_NOT_CALL_SPOOLER

An error occurred attempting to call the spooler validation routine. This error is not raised if the spooler is not installed.

PrfQueryProfileInt – Query Profile Integer

Remarks

This function returns an integer value from the profile. The call searches the file for a key matching the name specified by the *pszKey* parameter, under the application heading specified by the *pszApp* parameter. When an integer is stored as a text string using the *PrfWriteProfileString* function, for example, "123," the returned value is the number, 123. The call returns *lDefault* if the application-name or key-name pair cannot be found.

Note: The search is case-dependent.

Related Functions

- *PrfQueryProfileData*
- *PrfWriteProfileString*

Example Code

This example calls to search the user and system profiles for the integer value of key 'KEY' within the application 'APP' and return the value if found; if not found, 0 is returned.

```
#define INCL_WINSHELLDATA      /* Window Shell functions      */
#include <os2.h>

LONG lResult;                /* key value                */
HINI hini;                   /* initialization-file handle */
char pszApp[10];             /* application name          */
char pszKey[10];             /* key name                  */
LONG lDefault;               /* default return value      */

/* Both the user profile and system profile are searched */
hini = HINI_PROFILE;

/* specify application and key names */
strcpy(pszApp,"APP");
strcpy(pszKey,"KEY");

/* set default to 0 */
lDefault = 0;

lResult = PrfQueryProfileInt(hini, pszApp, pszKey, lDefault);
```

PrfQueryProfileSize — Query Profile Size

```
#define INCL_WINSHELLDATA /* Or use INCL_WIN or INCL_PM */
```

BOOL PrfQueryProfileSize (HINI hIni, PSZ pszApp, PSZ pszKey, PULONG pDataLen)

This function obtains the size in bytes of the value of a specified key for a specified application in the profile.

Parameters

hIni (HINI) — input
Initialization-file handle.

HINI_PROFILE Both the user profile and system profile are searched

HINI_USERPROFILE The user profile is searched

HINI_SYSTEMPROFILE The system profile is searched

Other Initialization-file handle.

pszApp (PSZ) — input
Application name.

The name of the application for which the profile data is required.

If the *pszApp* parameter is NULL, then the *pDataLen* parameter returns the length of the buffer required to hold the enumerated list of application names, as returned by the *PrfQueryProfileString* function when its *pszApp* parameter is NULL. In this case, the *pszKey* parameter is ignored.

pszKey (PSZ) — input
Key name.

The name of the key for which the size of the data is to be returned.

If the *pszKey* parameter is NULL, and if the *pszApp* parameter is not NULL, the *pDataLen* returns the length of the buffer required to hold the enumerated list of key names for that application name, as returned by the *PrfQueryProfileString* function when its *pszKey* parameter is NULL, and its *pszApp* parameter is not NULL.

pDataLen (PULONG) — output
Data length.

This parameter is the length of the value data related to the *pszKey* parameter. If an error occurs, this parameter is undefined.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from *WinGetLastError*

PMERR_INVALID_PARM

A parameter to the function contained invalid data.

PMERR_NOT_IN_IDX

The application name, key-name or program handle was not found.

PMERR_CAN_NOT_CALL_SPOOLER

An error occurred attempting to call the spooler validation routine. This error is not raised if the spooler is not installed.

PrfQueryProfileSize — Query Profile Size

Remarks

The *pszApp* parameter and *pszKey* parameter are case sensitive and must match the names stored in the file exactly. There is no case-independent searching.

This function can be used before using the *PrfQueryProfileString* call or the *PrfQueryProfileData* call, to allocate space for the returned data.

No distinction is made between data that is written using the *PrfWriteProfileData* function and the *PrfWriteProfileString* function.

Related Functions

- *PrfQueryProfileData*
- *PrfQueryProfileString*

Example Code

This example calls *PrfQueryProfileSize* to search the user and system profiles for the value of key 'KEY' within the application 'APP' and return the byte size of the value if found.

```
#define INCL_WINSHELLDATA      /* Window Shell functions      */
#include <os2.h>

BOOL fSuccess;                /* success indicator          */
HINI hini;                    /* initialization-file handle  */
char pszApp[10];              /* application name           */
char pszKey[10];              /* key name                   */
ULONG pDataLen;              /* data length                */

/* Both the user profile and system profile are searched */
hini = HINI_PROFILE;

/* specify application and key names */
strcpy(pszApp,"APP");
strcpy(pszKey,"KEY");

fSuccess = PrfQueryProfileSize(hini, pszApp, pszKey, &pDataLen);
```

PrfQueryProfileString — Query Profile String

```
#define INCL_WINSHELLDATA /* Or use INCL_WIN or INCL_PM */
```

**ULONG PrfQueryProfileString (HINI hini, PSZ pszApp, PSZ pszKey, PSZ pszDefault,
PSZ pszBuffer, ULONG cchBufferMax)**

This function retrieves a string from the specified profile.

Parameters

hini (HINI) — input
Initialization-file handle.

HINI_PROFILE Both the user profile and system profile are searched

HINI_USERPROFILE The user profile is searched

HINI_SYSTEMPROFILE The system profile is searched

Other Initialization-file handle.

pszApp (PSZ) — input
Application name.

The name of the application for which the profile data is required.

The search performed on the application name is always case-dependent. Names starting with the characters "PM_" are reserved for system use.

If this parameter is NULL, this function enumerates all the application names present in the profile and returns the names as a list in the *pszBuffer* parameter. Each application name is terminated with a NULL character and the last name is terminated with two successive NULL characters. In this instance, the *pullLength* parameter contains the total length of the list **excluding** the final NULL character.

pszKey (PSZ) — input
Key name.

The name of the key for which the profile data is returned.

The search on key name is always case-dependent.

If this parameter equals NULL, and if the *pszApp* parameter is not equal to NULL, this function enumerates all key names associated with the named application and returns the key names (not their values) as a list in the *pszBuffer* parameter. Each key name is terminated with a NULL character and the last name is terminated with two successive NULL characters. In this instance, the *pullLength* parameter contains the total length of the list **excluding** the final NULL character.

pszDefault (PSZ) — input
Default string.

The string that is returned in the *pszBuffer* parameter, if the key defined by the *pszKey* parameter cannot be found in the profile.

If the pointer to this parameter is passed as NULL, then nothing is copied into the *pszKey* parameter if the key cannot be found. *pullLength* is returned as 0 in this case.

pszBuffer (PSZ) — output
Profile string.

The text string obtained from the profile for the key defined by the *pszKey* parameter.

PrfQueryProfileString – Query Profile String

cchBufferMax (ULONG) – input
Maximum string length.

The maximum number of characters that can be put into the *pszBuffer* parameter, in bytes. If the data from the profile is longer than this, it is truncated.

Returns

String length returned.

The actual number of characters (including the null termination character) returned in the *pszBuffer* parameter, in bytes.

Possible returns from WinGetLastError

PMERR_INVALID_PARM	A parameter to the function contained invalid data.
PMERR_BUFFER_TOO_SMALL	The supplied buffer was not large enough for the data to be returned.
PMERR_NOT_IN_IDX	The application name, key-name or program handle was not found.
PMERR_CAN_NOT_CALL_SPOOLER	An error occurred attempting to call the spooler validation routine. This error is not raised if the spooler is not installed.
PMERR_INVALID_ASCII	The profile string is not a valid zero-terminated string.

Remarks

The call searches the profile for a key matching the name specified by the *pszKey* parameter under the application heading specified by the *pszApp* parameter. If the key is found, the corresponding string is copied. If the key does not exist, the default character string, specified by the *pszDefault* parameter, is copied.

If the enumerated application names exceed the available buffer space, the enumerated names are truncated, the enumerated list is not terminated with 2 bytes of zeros, and the *pullLength* parameter is set to the number of bytes copied into the *pszBuffer* parameter. In this instance, the *pszKey* parameter is ignored.

Note: If the enumeration cannot be performed for any reason, the default character string is *not* copied.

This function returns the length of the list, up to, but not including, the final null. If the enumerated key names exceed the available buffer space, the enumerated names are truncated, the enumerated list is not terminated with 2 bytes of zeros, and the *pullLength* parameter is set to the number of bytes copied into the *pszBuffer* parameter.

This function is case-dependent; thus the strings in the *pszApp* parameter and the *pszKey* parameter must match exactly. This avoids any code-page dependency. The application storing the data must do any case-independent matching.

The enumeration call does not distinguish between data written with the *PrfWriteProfileString* function and the *PrfWriteProfileData* function.

Related Functions

- *PrfWriteProfileString*

PrfQueryProfileString — Query Profile String

Example Code

PrfQueryProfileString is issued twice to obtain the names of the default printer, the default presentation driver, and the queue associated with the printer. If any of these requests fails, the default values already defined in DEVOPENSTRUC are used.

```
#define INCL_WINSHELLDATA
#include <OS2.H>
char szTemp[80];
char szBuff[257];
PCH ptscan;

DEVOPENSTRUC dopPrinter = {"LPT1Q",
    (PSZ)"IBM4201",
    0L,
    (PSZ)"PM_Q_STD",
    0L, 0L, 0L, 0L, 0L};

if (PrfQueryProfileString(HINI_PROFILE,
    (PSZ)"PM_SPOOLER",
    (PSZ)"PRINTER",
    NULL,
    (PSZ)szTemp,
    (LONG)sizeof(szTemp)
)){
    szTemp[strlen(szTemp)-1] = 0;
    if (PrfQueryProfileString(HINI_PROFILE,
        (PSZ)"PM_SPOOLER_PRINTER",
        (PSZ)szTemp,
        NULL,
        (PSZ)szBuff,
        (LONG)sizeof(szBuff)
    )){
/*  char * strchr( const char *, int ); */
        ptscan = (PCH)strchr(szBuff, ';');
        ptscan++;
        ptscan = (PCH)strchr(ptscan, (INT)';');
        ptscan++;
        *(ptscan + strcspn(ptscan, ".,;")) = 0;
        dopPrinter.pszLogAddress = ptscan;

        ptscan = (PCH)strchr(szBuff, (INT)';');
        ptscan++;
        *(ptscan + strcspn(ptscan, ".,;")) = 0;
        dopPrinter.pszDriverName = ptscan;
    }
}
```

PrfReset — Reset Presentation Manager

```
#define INCL_WINSHELLDATA /* Or use INCL_WIN or INCL_PM */
```

BOOL PrfReset (HAB hab, PPRFPROFILE pprfproProfile)

This function defines which files are to be used as the user and system profiles.

Parameters

hab (HAB) – input
Anchor-block handle.

pprfproProfile (PRFPROFILE) – input
Profile-names structure.

This contains the names of the files to be used as the new Presentation Manager (PM) profile files. Any valid file names can be used. A name that is not already fully qualified is taken to refer to the current directory.

If the user profile file does not exist, a new file is created.

The name of the system profile cannot be changed. It must be the name of the current system profile as returned by PrfQueryProfile.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_OPENING_INI_FILE

Unable to open initialization file (due to lack of disk space for example).

Remarks

This function causes the workstation to use different profiles. When the workstation is initialized, the names of the user and system profiles are taken from the PROTSHELL statement specified in CONFIG.SYS. PrfReset allows the profiles to be changed during operation of the workstation, for example by a logon application controlling multiple consecutive users of the system.

After the PrfReset function completes, the system has a new set of preferences (for example screen colors), a new start-up list, and new spooler parameters.

The PrfReset function broadcasts the PL_ALTERED message, which must be processed by all applications that read their default settings from the user or system profiles.

Note: This will only change the default system values in the ini file. It is up to the applications to read the new default settings and reset them to their new values.

For example, consider logon applications. On receipt of a PL_ALTERED message, they should carry out the following:

- Read the new color settings from the new profiles, and set the new screen colors (and palettes) which should be refreshed.

PrfReset — Reset Presentation Manager

- Set the country information, for example the date and time format, which is read from the new profiles.
- Other preferences, for example, those that affect the operations of the alarm and the mouse, should also update with the new settings held in the new profiles.

This function requires the existence of a message queue.

Related Functions

- PrfQueryProfile

Related Messages

- PL_ALTERED

Example Code

This function defines which files are to be used as the user and system profiles.

```
#define INCL_WINSHELLDATA
#include <OS2.H>
HAB hab;
char userpro[] = "profile.ini";
PRFPROFILE profile;

PrfQueryProfile(hab, &profile); /* get the system profile name */
                               /* which cannot be changed.    */

profile.pszUserName = userpro;
profile.cchSysName = sizeof(profile.pszUserName);

PrfReset (hab, &profile);
```

PrfWriteProfileData – Write Profile Data

```
#define INCL_WINSHELLDATA /* Or use INCL_WIN or INCL_PM */
```

```
BOOL PrfWriteProfileData (HINI hIni, PSZ pszApp, PSZ pszKey, PVOID pData,  
                          ULONG cchDataLen)
```

This function writes a string of binary data into the specified profile.

Parameters

hIni (HINI) – input
Initialization-file handle.

HINI_PROFILE User profile

HINI_USERPROFILE User profile

HINI_SYSTEMPROFILE System profile

Other Initialization-file handle.

pszApp (PSZ) – input
Application name.

The case-dependent name of the application for which profile data is to be written. Names starting with the characters "PM_" are reserved for system use.

pszKey (PSZ) – input
Key name.

The case-dependent name of the key for which profile data is to be written.

This parameter can be NULL in which case *all* the *pszKey* or *pData* pairs associated with *pszApp* are deleted.

pData (PVOID) – input
Value data.

This is the value of the *pszKey* or *pData* pair that is written to the profile. It is *not* zero-terminated, and its length is given by the *cchDataLen* parameter.

If this parameter is NULL, the string associated with the *pszKey* parameter is deleted.

cchDataLen (ULONG) – input
Size of value data.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INVALID_PARM

A parameter to the function contained invalid data.

PMERR_CAN_NOT_CALL_SPOOLER

An error occurred attempting to call the spooler validation routine. This error is not raised if the spooler is not installed.

PrfWriteProfileData —

Write Profile Data

Remarks

Because of the binary nature of the data, the input data is not zero-terminated. The length provided is the only way to identify the length of the data.

Related Functions

- PrfQueryProfileSize

Example Code

This function deletes the profile data associated with application sample.exe

```
#define INCL_WINSHELLDATA
#include <OS2.H>
HAB hab;

PrfWriteProfileData(HINI_USERPROFILE,
                    "sample",          /* application. */
                    NULL,
                    NULL,
                    0L);
```

PrfWriteProfileString — Write Profile String

```
#define INCL_WINSHELLDATA /* Or use INCL_WIN or INCL_PM */
```

BOOL PrfWriteProfileString (HINI hIni, PSZ pszApp, PSZ pszKey, PSZ pszData)

This function writes a string of character data into the specified profile.

Parameters

hIni (HINI) — input
Initialization-file handle.

HINI_PROFILE User profile

HINI_USERPROFILE User profile

HINI_SYSTEMPROFILE System profile

Other Initialization-file handle.

pszApp (PSZ) — input
Application name.

The case-dependent name of the application for which profile data is to be written. Names starting with the characters "PM_" are reserved for system use.

pszKey (PSZ) — input
Key name.

The case-dependent name of the key for which profile data is to be written.

This parameter can be NULL, in which case *all* the *pszKey* or *pszData* pairs associated with the *pszApp* parameter are deleted.

pszData (PSZ) — input
Text string.

This is the value of the *pszKey* or *pszData* pair that is written to the profile.

If this parameter is NULL, the string associated with the *pszKey* is deleted (that is, the entry is deleted).

If this parameter is not NULL, the string is used as the value of the *pszKey* or *pszData* pair, even if the string has zero length.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INVALID_PARM

A parameter to the function contained invalid data.

PMERR_CAN_NOT_CALL_SPOOLER

An error occurred attempting to call the spooler validation routine. This error is not raised if the spooler is not installed.

PrfWriteProfileString —

Write Profile String

Remarks

If there is no application field in the file that matches the *pszApp*, a new application field is created before the *pszKey* or *pszData* entry is made.

If the key name does not exist for the application, a new *pszKey* or *pszData* entry is created for that application. If the *pszKey* already exists in the file, the existing value is overwritten.

Related Functions

- PrfQueryProfileString

Example Code

This function deletes the profile string associated with application sample.exe

```
#define INCL_WINSHELLDATA
#include <OS2.H>
HAB hab;
```

```
PrfWriteProfileString(HINI_USERPROFILE,
                     "sample",          /* application. */
                     NULL,
                     NULL);
```

Chapter 7. Spooler Functions

The following table shows how all of the Spooler functions are related within functional areas. The functions are in alphabetic order within these areas.

C Name	C Name
Control	
SplControlDevice	SplEnumQueueProcessor
SplCopyJob	SplHoldJob
SplCreateDevice	SplHoldQueue
SplCreateQueue	SplPurgeQueue
SplDeleteDevice	SplQueryDevice
SplDeleteJob	SplQueryJob
SplDeleteQueue	SplQueryQueue
SplEnumDevice	SplReleaseJob
SplEnumDriver	SplReleaseQueue
SplEnumJob	SplSetDevice
SplEnumPort	SplSetJob
SplEnumPrinter	SplSetQueue
SplEnumQueue	
Job Submission	
SplQmAbort	SplQmOpen
SplQmAbortDoc	SplQmStartDoc
SplQmClose	SplQmWrite
SplQmEndDoc	

SplControlDevice – Spooler Control Device

```
#define INCL_SPL /* Or use INCL_PM */
```

SPLERR SplControlDevice (PSZ pszComputerName, PSZ pszPortName, ULONG ulControl)

This function cancels, holds, continues, or restarts a print device.

Parameters

pszComputerName (PSZ) – input

Name of computer where print device is to be controlled.

A NULL string specifies the local workstation.

pszPortName (PSZ) – input

Port name.

ulControl (ULONG) – input

Operation to perform.

PRD_DELETE Delete current print job

PRD_PAUSE Pause printing

PRD_CONT Continue paused print job

PRD_RESTART Restart print job.

Returns

NO_ERROR (0)

No errors occurred.

ERROR_NOT_SUPPORTED (50)

This request is not supported by the network.

ERROR_BAD_NETPATH (53)

The network path cannot be located.

NERR_NetNotStarted (2102)

The network program is not started.

NERR_DestNotFound (2152)

The print device cannot be found.

NERR_DestIdle (2158)

This print device is idle and cannot accept control operations.

NERR_DestInvalidOp (2159)

This print device request contains an invalid control function.

NERR_ProcNoRespond (2160)

The queue processor is not responding.

NERR_SpoolerNotLoaded (2161)

The spooler is not running.

NERR_InvalidComputer (2351)

The computer name is invalid.

Remarks

A paused print device cannot accept new print jobs.

If PRD_DELETE is attempted when there is no current print job, NERR_DestIdle (2158) is returned.

To control jobs on a remote server requires administrator privilege.

Related Functions

- SplEnumDevice
- SplQueryDevice

Example Code

This sample code demonstrates the result of various actions that can be performed on the print device by this function call. At the command line, a print device name is entered along with an action code.

```
#define INCL_SPL
#define INCL_SPLDOSPRINT
#include <os2.h>
#include <stdio.h>      /* for printf function */
#include <neterr.h>     /* for error codes */

INT main (argc, argv)
    INT argc;
    CHAR *argv[];
{
    SPLERR splerr ;
    ULONG ulControl=0L ;
    PSZ   pszComputerName = NULL ;
    PSZ   pszPrintDeviceName ;

    /* Input a Print Device Name and an Action Code on the command line */
    if (argc != 3)
    {
        printf("Syntax is: qcontrol PrintDeviceName ActionCode \n");
        printf("Action codes are: D-Delete, P-Pause, C-Continue, R-Restart\n\n");
        DosExit( EXIT_PROCESS , 0 ) ;
    }
    /* Get the print device name from the first input parameter. */
    pszPrintDeviceName = argv[1];

    /* Get the action code from the second input parameter. */
    switch (argv[2][0])
    {
        case 'D':
            ulControl = PRD_DELETE ;
            break;
        case 'P':
            ulControl = PRD_PAUSE ;
            break;
        case 'C':
            ulControl = PRD_CONT ;
            break;
        case 'R':
            ulControl = PRD_RESTART ;
            break;
        default:
            printf("Invalid code\n");
            DosExit( EXIT_PROCESS , 0 ) ;
    }
    /* Call the function with the parameters obtained from the command line. */
    splerr = SplControlDevice(pszComputerName, pszPrintDeviceName, ulControl);

    /* If there is an error returned, print it. */
    if (splerr != 0L)
    {
        switch (splerr)
        {
```

SplControlDevice — Spooler Control Device

```
    case NERR_DestNotFound :
        printf("Destination does not exist.\n");
        break;
    case NERR_DestIdle:
        printf("This print device is idle - can't do control ops. \n");
        break;
    default:
        printf("Errorcode = %ld\n",splerr);
    }
} else {
    printf("The print job operation was performed.\n\n");
}
DosExit( EXIT_PROCESS , 0 ) ;
return (splerr) ;
}
```

SplCopyJob – Spooler Copy Job

```
#define INCL_SPL /* Or use INCL_PM */
```

```
SPLERR SplCopyJob (PSZ pszSrcComputerName, PSZ pszSrcQueueName, ULONG ulSrcJob,  
PSZ pszTrgComputerName, PSZ pszTrgQueueName,  
PULONG pulTrgJob)
```

This function copies a job in a print queue.

Parameters

- pszSrcComputerName (PSZ)** – input
Name of computer where job is to be copied from.
A NULL string specifies the local workstation.
- pszSrcQueueName (PSZ)** – input
Name of queue where job is to be copied from.
- ulSrcJob (ULONG)** – input
Source Job identification number.
- pszTrgComputerName (PSZ)** – input
Name of computer where job is to be copied to.
A NULL string specifies the local workstation.
- pszTrgQueueName (PSZ)** – input
Name of queue where job is to be copied to.
A NULL string specifies the same queue as the original job.
- pulTrgJob (PULONG)** – output
Job identification number of new job.

Returns

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_QNotFound (2150)	The printer queue does not exist.
NERR_JobNotFound (2151)	The print job does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

Currently there is a restriction that a job can only be copied onto the same queue (and computer) as the original job.

SplCopyJob – Spooler Copy Job

Related Functions

- SplEnumJob
- SplEnumQueue
- SplQueryJob
- SplQueryQueue

Example Code

This sample code will make a duplicate copy of the jobid that is entered at the prompt. Presently, there is a restriction that the job can only be duplicated on the same computer/queue; for example, a local job.

```
#define INCL_SPL
#include <os2.h>
#include <stdio.h>      /* for printf function */
#include <stdlib.h>     /* for atoi function */

INT main (argc, argv)
    INT argc;
    CHAR *argv[];
{
    SPLERR splerr;
    ULONG ulSrcJob, ulTrgJob;
    PSZ    pszSrcComputerName, pszTrgComputerName;
    PSZ    pszSrcQueueName, pszTrgQueueName;

    if (argc != 2)
    {
        printf("Command is: copyjob JOBID\n");
        DosExit( EXIT_PROCESS, 0 );
    }
    pszSrcComputerName = (PSZ)NULL;

    /* The only valid values at present for these three parameters is NULL */
    pszSrcQueueName = (PSZ)NULL;
    pszTrgComputerName = (PSZ)NULL;
    pszTrgQueueName = (PSZ)NULL;

    /* Convert input parameter to a ULONG */
    ulSrcJob = atoi ( argv[1] );

    if (splerr = SplCopyJob(pszSrcComputerName, pszSrcQueueName, ulSrcJob,
                           pszTrgComputerName, pszTrgQueueName, &ulTrgJob))
    {
        printf("Return code SplCopyJob = %d\n", splerr);
    }
    else
    {
        printf("New job ID is %d\n", ulTrgJob);
    }
    DosExit( EXIT_PROCESS, 0 );
    return (splerr);
} /* end main */
```

SplCreateDevice — Spooler Create Device

```
#define INCL_SPL /* Or use INCL_PM */
```

**SPLERR SplCreateDevice (PSZ pszComputerName, ULONG ulLevel, PVOID pBuf,
ULONG cbBuf)**

This function establishes a print device on the local workstation or a remote server.

Parameters

- pszComputerName (PSZ)** — input
Name of computer where print device is to be added.
A NULL string specifies the local workstation.
- ulLevel (ULONG)** — input
Level of detail provided.
This must be 3.
- pBuf (PVOID)** — input
Data structure.
- cbBuf (ULONG)** — input
Size, in bytes, of data structure.

Returns

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_NAME (123)	The computer name is invalid.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_BufTooSmall (2123)	The API return buffer is too small.
NERR_DestExists (2153)	The print device already exists.
NERR_DestNoRoom (2157)	The maximum number of print devices has been reached.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_DestInvalidState (2162)	This operation cannot be performed on the print device.
NERR_SpoolNoMemory (2165)	A spooler memory allocation failure occurred.
NERR_DriverNotFound (2166)	The device driver does not exist.
NERR_BadDev (2341)	The device is already in use as a communications device.
NERR_InvalidComputer (2351)	The computer name is invalid.

SplCreateDevice — Spooler Create Device

Remarks

The result of this function is the creation of a new print device definition.

The printer is set up to print on the logical address (port) defined by *pszLogAddr* in PRDINFO3. If *pszLogAddr* is NULL, the print device definition is created but is not connected to any logical address. In this case no printing can occur on that print device or from any print queue connected only to that print device. If a logical address is specified, it must already be defined in the PM_SPOOLER_PORTS section of the initialization file.

Note: To change the connection between a print device and a port, use SplSetDevice.

The maximum length for a print device name is 32 characters. The use of a longer name results in ERROR_INVALID_NAME (123).

All device drivers and queues specified with the print device must already be defined to the spooler.

To add a remote print device requires administrator privilege.

Related Functions

- SplDeleteDevice
- SplEnumDevice
- SplEnumDriver
- SplEnumPort

SplCreateDevice – Spooler Create Device

Example Code

This sample code creates a PRDINFO3 structure with dummy parameters. This structure is then used to call SplCreateDevice to establish a print device on a local workstation.

```
#define INCL_BASE
#define INCL_DOSMEMMGR
#define INCL_SPL
#define INCL_SPLDOSPRINT

#include <os2.h>
#include <stdio.h>    /* for printf function */
#include <string.h>    /* for strcpy function */

INT main (argc, argv)
    INT argc;
    CHAR *argv[];
{
    ULONG splerr ;
    ULONG cbBuf;
    ULONG ulLevel ;
    PSZ   pszComputerName ;
    PSZ   pszPrintDeviceName ;
    PRDINFO3 prd3 ;

    if (argc != 2)
    {
        printf("Syntax: sdcr DeviceName \n");
        DosExit( EXIT_PROCESS , 0 ) ;
    }
    /* We are going to create a print device on the local workstation. */
    pszComputerName = (PSZ)NULL ;

    /* Get the name from the command line. */
    pszPrintDeviceName = argv[1];

    /* Level 3 is valid. We will use level 3. */
    ulLevel = 3;

    /* Get size of buffer needed for a PRDINFO3 structure. */
    cbBuf = sizeof(PRDINFO3);

    /* Set up the structure with dummy parameters. */
    strcpy( prd3.pszPrinterName , pszPrintDeviceName);
    prd3.pszUserName= "A. Best";
    prd3.pszLogAddr="LPT1Q";
    prd3.uJobId=0;
    prd3.pszComment= "Test comment";
    prd3.pszDrivers = "IBMNULL";
    prd3.usTimeOut = 777;

    /* Make the call. */
    splerr = SplCreateDevice(pszComputerName, ulLevel,
                            &prd3, cbBuf);

    /* Print out the results. */
    if (splerr == NO_ERROR)
        printf("The device was successfully created.");
    else
        printf("SplCreateDevice Error=%ld, cbNeeded=%ld\n",
               splerr, cbBuf) ;

    DosExit( EXIT_PROCESS , 0 ) ;
    return (splerr);
}
```


SplCreateQueue — Spooler Create Queue

```
#define INCL_SPL /* Or use INCL_PM */
```

**SPLERR SplCreateQueue (PSZ pszComputerName, ULONG ulLevel, PVOID pbBuf,
ULONG cbBuf)**

This function creates a new print queue on the local workstation or on a remote server. A remote server setup requires the LAN Requester and Server software.

Parameters

- pszComputerName (PSZ)** — input
Name of computer where queue is to be created.
A NULL string specifies a local workstation.
- ulLevel (ULONG)** — input
Level of detail provided.
This must be 3 or 6.
- pbBuf (PVOID)** — input
Data structure.
- cbBuf (ULONG)** — input
Size, in bytes, of data structure.

Returns

NO_ERROR (0)	No errors occurred.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_NAME (123)	The computer name is invalid.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_RedirectedPath (2117)	The operation is invalid on a redirected resource.
NERR_BufTooSmall (2123)	The API return buffer is too small.
NERR_DestNotFound (2152)	The printer destination cannot be found.
NERR_QExists (2154)	The printer queue already exists.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_DestInvalidState (2162)	This operation cannot be performed on the print destination in its current state.
NERR_SpoolNoMemory (2165)	A spooler memory allocation failure occurred.
NERR_DriverNotFound (2166)	The device driver does not exist.
NERR_DataTypeInvalid (2167)	The data type is not supported by the queue processor.
NERR_ProcNotFound (2168)	The queue processor is not installed.
NERR_BadDev (2341)	The requested device is invalid.
NERR_CommDevInUse (2343)	This device is already in use as a communications device.
NERR_InvalidComputer (2351)	The computer name is invalid.

SplCreateQueue – Spooler Create Queue

Remarks

To create a queue on a remote server requires administrator privilege.

Applications wanting to create print queues should use the level 3 or level 6 call with a PRQINFO3 or PRQINFO6 data structure. The following fields are required in PRQINFO3 or PRQINFO6:

pszName
uPriority
uStartTime
uUntilTime
pszSepFile
pszParms
pszPrinters
pszDriverName
pDriverData.

If a queue of the name specified in *pszName* already exists on *pszComputerName*, the call fails unless the queue is marked for deletion. In this case, the queue is not deleted, and the creation fields are used to perform a SplSetQueue function on the queue.

If *pszPrinters* is NULL, the queue is created but not connected to any printer.

The queue that is created has a status of PRQ3_PENDING even if the queue is not connected to a printer.

pszDriverName can be a NULL string, in which case *pDriverData* is ignored. Otherwise *pszDriverName* must refer to the name of a device driver that is already defined in the initialization file (for example, "IBM4019").

SplCreateQueue — Spooler Create Queue

Related Functions

- SplDeleteQueue
- SplEnumDevice
- SplEnumDriver
- SplEnumQueueProcessor

Example Code

This sample code creates a queue on the local workstation. The queue is created with dummy parameters. The name is entered at the command line.

```
#define INCL_BASE
#define INCL_SPL
#define INCL_SPLDOSPRINT

#include <os2.h>
#include <stdio.h>
#include <string.h>

INT main (argc, argv )
    INT argc;
    CHAR *argv[];
{
    ULONG splerr ;
    ULONG cbBuf;
    ULONG ulLevel ;
    PSZ   pszComputerName ;
    PSZ   pszQueueName ;
    PRQINF03 prq3 ;

    if (argc != 2)
    {
        printf("Syntax: sqcrt QueueName \n");
        DosExit( EXIT_PROCESS , 0 ) ;
    }

    pszComputerName = (PSZ)NULL ;
    ulLevel = 3L;

    /* Get the queue name from the argument entered at */
    /* the command line. */
    pszQueueName = argv[1];

    /* Determine the size of the needed buffer. */
    cbBuf = sizeof(PRQINF03);

    /* Set up the structure with some dummy parameters. */
    strcpy( prq3.pszName , pszQueueName);
    prq3.uPriority=7;
    prq3.uStartTime=77;
    prq3.uUntilTime=777;
    prq3.pszSepFile="a:\\best\\example.sep";
    prq3.pszParms=NULL;
    prq3.pszPrinters=NULL;
    prq3.pszDriverName=NULL;
    prq3.pDriverData=NULL;
```

SplCreateQueue — Spooler Create Queue

```
/* Make the call with the proper parameters. */
splerr = SplCreateQueue(pszComputerName, ulLevel,
                        &prq3, cbBuf);

/* Print out the error return code and some other information. */
printf("SplCreateQueue Error=%ld, cbNeeded=%ld\n",
       splerr, cbBuf);

DosExit( EXIT_PROCESS , 0 );
return (splerr);
}
```

SplDeleteDevice – Spooler Delete Device

```
#define INCL_SPL /* Or use INCL_PM */
```

SPLERR SplDeleteDevice (PSZ pszComputerName, PSZ pszPrintDeviceName)

This function deletes a print device.

Parameters

pszComputerName (PSZ) – input

Name of computer where print device is to be deleted.

A NULL string specifies the local workstation.

pszPrintDeviceName (PSZ) – input

Name of Print Device.

Returns

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_DestNotFound (2152)	The print device cannot be found.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_DestInvalidState (2162)	This operation cannot be performed on the print device.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

If the print device is currently printing a job, SplDeleteDevice fails and returns NERR_DestInvalidState (2162).

To delete a print device on a remote server requires administrator privilege.

Related Functions

- SplCreateDevice
- SplEnumDevice

SplDeleteDevice – Spooler Delete Device

Example Code

This sample code will delete the print device whose name is entered at the prompt.

```
#define INCL_BASE
#define INCL_DOSMEMMGR
#define INCL_SPL
#define INCL_SPLDOSPRINT

#include <os2.h>
#include <stdio.h>
#include <neterr.h>

INT main (argc, argv)
    INT argc;
    CHAR *argv[];
{
    SPLERR splerr= 0L;
    PSZ    pszComputerName ;
    PSZ    pszPrintDeviceName ;

    /* Check that the parameters were entered at the command line.          */
    if (argc != 2)
    {
        printf("Syntax: sddel PrintDeviceName \n");
        DosExit( EXIT_PROCESS , 0 ) ;
    }
    /* Computer name of NULL indicates the local computer.                  */
    pszComputerName = (PSZ)NULL ;

    /* Set the PrintDeviceName to the value entered at the command line.    */
    pszPrintDeviceName = argv[1];

    /* Make the call and print out the return code.                        */
    splerr=SplDeleteDevice(pszComputerName, pszPrintDeviceName);
    switch (splerr)
    {
        case NO_ERROR:
            printf("Print Device %s was deleted.\n",pszPrintDeviceName);
            break;
        case NERR_DestNotFound :
            printf("Destination does not exist.\n");
            break;
        case NERR_DestInvalidState:
            printf("This operation can't be performed on the print device.\n");
            break;
        case NERR_SpoolerNotLoaded:
            printf("The Spooler is not running.\n");
            break;
        default:
            printf("SplDeleteDevice Errorcode = %ld\n",splerr);
    } /* endswitch */
    DosExit( EXIT_PROCESS , 0 ) ;
    return (splerr) ;
}
```

SplDeleteJob – Spooler Delete Job

```
#define INCL_SPL /* Or use INCL_PM */
```

SPLERR SplDeleteJob (PSZ pszComputerName, PSZ pszQueueName, ULONG ulJob)

This function deletes a job from a print queue.

Parameters

pszComputerName (PSZ) – input
Name of computer where job is to be deleted.
A NULL string specifies the local workstation.

pszQueueName (PSZ) – input
Queue Name.

ulJob (ULONG) – input
Job identification number.

Returns

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_JobNotFound (2151)	The print job does not exist.
NERR_ProcNoRespond (2160)	The queue processor is not responding.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

It is possible to delete a job that is currently printing.

If the print queue on which the print job is submitted is pending deletion (following a SplDeleteQueue call), and the print job is the last in the queue, this function has the additional effect of deleting the queue.

A user with administrator privilege can delete any job.

A job created locally can be deleted locally regardless of user privilege level, but can be deleted remotely only by an administrator.

A remote job can be deleted by a user without administrator privilege only if the username of the person initiating the request is the same as the username of the person who created the job.

Related Functions

- SplCopyJob
- SplEnumJob
- SplQueryJob

Example Code

This sample code will delete the job id that is entered at the prompt.

```
#define INCL_BASE
#define INCL_SPL
#include <os2.h>
#include <stdio.h>      /* for printf function */
#include <neterr.h>      /* for error codes */
#include <stdlib.h>      /* for atoi function */

INT main (argc, argv)
    INT argc;
    CHAR *argv[];
{
    SPLERR splerr ;
    ULONG ulJob ;
    PSZ   pszComputerName = NULL ;
    PSZ   pszQueueName = NULL ;

    /* Get job id from the input argument. */
    ulJob = atoi(argv[1]);

    /* Call the function to do the delete. If an error is
    /* returned, print it. */
    splerr = SplDeleteJob( pszComputerName, pszQueueName, ulJob);

    if (splerr != NO_ERROR)
    {
        switch (splerr)
        {
            case NERR_JobNotFound :
                printf("Job does not exist.\n");
                break;
            case NERR_JobInvalidState:
                printf("This operation can't be performed on the print job.\n");
                break;
            default:
                printf("Errorcode = %ld\n",splerr);
        } /* endswitch */
    }
    else
    {
        printf("Job %d was deleted.\n",ulJob);
    } /* endif */
    DosExit( EXIT_PROCESS , 0 ) ;
    return (splerr);
}
```


SplDeleteQueue — Spooler Delete Queue

```
#define INCL_SPL /* Or use INCL_PM */
```

SPLERR SplDeleteQueue (PSZ pszComputerName, PSZ pszQueueName)

This function deletes a print queue from the spooler.

Parameters

pszComputerName (PSZ) — input

Name of computer where queue is to be deleted.

A NULL string specifies the local workstation.

pszQueueName (PSZ) — input

Queue name.

Returns

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_QNotFound (2150)	The printer queue does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_QInvalidState (2163)	This operation cannot be performed on the print queue.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

If there are print jobs in the queue, SplDeleteQueue marks the queue PRQ3_PENDING. No further jobs can then be added to the queue, which is deleted when all jobs are printed. A queue marked PRQ3_PENDING can be held, and jobs in the queue can be held, restarted, and repeated.

If a queue is held and there are jobs on the queue, a SplDeleteQueue function fails with NERR_QInvalidState (2163).

To delete a queue on a remote server requires administrator privilege on the remote server.

Related Functions

- SplCreateQueue
- SplEnumQueue
- SplQueryQueue

SplDeleteQueue – Spooler Delete Queue

Example Code

This sample code will delete the queue name that is entered at the prompt.

```
#define INCL_SPL
#include <os2.h>
#include <stdio.h>      /* for printf function */
#include <neterr.h>     /* for error codes */

INT main (argc, argv)
    INT argc;
    CHAR *argv[];
{
    SPLERR splerr ;
    PSZ   pszComputerName = NULL ;
    PSZ   pszQueueName ;

    /* Get queue name from the input argument */
    pszQueueName = argv[1];

    /* Call the function to do the delete. If an error is returned, print it.
    */
    splerr=SplDeleteQueue(pszComputerName, pszQueueName);

    if (splerr != 0L)
    {
        switch (splerr)
        {
            case NERR_QNotFound :
                printf("Queue does not exist.\n");
                break;
            case NERR_QInvalidState:
                printf("This operation can't be performed on the print queue.\n");
                break;
            default:
                printf("Errorcode = %ld\n",splerr);
        } /* endswitch */
    }
    else
    {
        printf("Queue %s was deleted.\n",pszQueueName);
    } /* endif */
    DosExit( EXIT_PROCESS , 0 ) ;
    return (splerr);
}
```

SplEnumDevice — Spooler Enumerate Device

```
#define INCL_SPL /* Or use INCL_PM */
```

SPLERR SplEnumDevice (PSZ pszComputerName, ULONG ulLevel, PVOID pBuf,
ULONG cbBuf, PULONG pcReturned, PULONG pcTotal,
PULONG pcbNeeded, PVOID pReserved)

This function lists print device on a server, optionally supplying status information.

Parameters

pszComputerName (PSZ) — input
Name of computer where print devices are to be listed.
A NULL string specifies the local workstation.

ulLevel (ULONG) — input
Level of detail required.
This must be 0, 2 or 3.

pBuf (PVOID) — output
Buffer.

cbBuf (ULONG) — input
Size, in bytes, of Buffer.

pcReturned (PULONG) — output
Number of entries returned.

pcTotal (PULONG) — output
Number of entries available.

pcbNeeded (PULONG) — output
Size in bytes of available information.
A value of 0 specifies that the size is not known.

pReserved (PVOID) — output
Reserved.
This must be NULL.

Returns

NO_ERROR (0)	No errors occurred.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
ERROR_MORE_DATA (234)	Additional data is available.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

SplEnumDevice – Spooler Enumerate Device

Remarks

The buffer contents on return are:

<i>ulLevel</i>	Buffer Contents
0	An array of port names of type PSZ.
2	An array of print device names of type PSZ.
3	An array of PRDINFO3 structures.

If no job is printing on the print device, bits 2 – 11 of *fsStatus* in the PRDINFO3 data structure are meaningless.

Related Functions

- SplCreateDevice
- SplDeleteDevice

Example Code

This sample code enumerates all the devices on the local workstation. It then prints out the information.

```
#define INCL_BASE
#define INCL_DOSMEMMGR
#define INCL_SPL
#define INCL_SPLDOSPRINT

#include <os2.h>
#include <stdio.h>
#include <neterr.h>

INT main ()
{
    ULONG cbBuf ;
    ULONG cTotal;
    ULONG cReturned ;
    ULONG cbNeeded ;
    ULONG ulLevel = 3L;
    ULONG i ;
    SPLERR splerr ;
    PSZ pszComputerName ;
    PBYTE pBuf ;
    PPRDINFO3 pprd3 ;

    pszComputerName = (PSZ)NULL ;

    /* Make the call with cbBuf = 0 so that you will get the size of the */
    /* buffer needed returned in cbNeeded. */
    splerr = SplEnumDevice(pszComputerName, ulLevel, pBuf, 0L, /* cbBuf */
                          &cReturned, &cTotal, &cbNeeded,
                          NULL) ;

    /* Only continue if the error codes ERROR_MORE_DATA or */
    /* NERR_BufTooSmall are returned. */
    if (splerr == ERROR_MORE_DATA || splerr == NERR_BufTooSmall)
    {
        /* Allocate memory for the buffer that will hold the returning info. */
        if (!DosAllocMem( &pBuf, cbNeeded,
                        PAG_READ|PAG_WRITE|PAG_COMMIT) )
        {
            cbBuf = cbNeeded ;

            /* Make call again with the proper buffer size. */
            splerr = SplEnumDevice(pszComputerName, ulLevel, pBuf, cbBuf,
```

SplEnumDevice — Spooler Enumerate Device

```

                                &cReturned, &cTotal,
                                &cbNeeded, NULL) ;

/* If no errors, print out the buffer information.                */
if (splerr == NO_ERROR)
{
    for (i=0; i < cReturned ; i++)
    {
        /* Each time through the loop increase the pointer.        */
        pprd3 = (PPRDINFO3)pBuf+i ;
        printf("Device info:pszPrinterName   - %s\n",
                pprd3->pszPrinterName) ;
        printf("  pszUserName - %s\n", pprd3->pszUserName);
        printf("  pszLogAddr  - %s\n", pprd3->pszLogAddr);
        printf("  uJobId      - %d  fsStatus - %X\n",
                pprd3->uJobId , pprd3->fsStatus);
        printf("  pszStatus   - %s\n", pprd3->pszStatus);
        printf("  pszComment  - %s\n", pprd3->pszComment);
        printf("  pszDrivers  - %s\n", pprd3->pszDrivers);
        printf("  time       - %d  usTimeOut - %X\n",
                pprd3->time , pprd3->usTimeOut);
    }
    DosFreeMem(pBuf) ;
}
} /* end if */
else
{
    printf("SplEnumDevice splerr=%ld, cTotal=%ld, cReturned=%ld,
cbNeeded=%ld\n",
                                splerr, cTotal, cReturned, cbNeeded) ;
}
DosExit( EXIT_PROCESS , 0 ) ;
return(splerr);
} /* end main */

```

SplEnumDriver – Spooler Enumerate Driver

```
#define INCL_SPL /* Or use INCL_PM */
```

```
SPLERR SplEnumDriver (PSZ pszComputerName, ULONG ulLevel, PVOID pBuf, ULONG cbBuf,  
PULONG pcReturned, PULONG pcTotal, PULONG pcbNeeded,  
PVOID pReserved)
```

This function lists printer presentation drivers on the local workstation or on a remote server.

Parameters

pszComputerName (PSZ) – input
Name of computer where queues are to be listed.
A NULL string specifies the local workstation.

ulLevel (ULONG) – input
Level of detail.
The level of detail required. This must be 0.

pBuf (PVOID) – output
Buffer.

cbBuf (ULONG) – input
Size, in bytes, of Buffer.

pcReturned (PULONG) – output
Number of entries returned.

pcTotal (PULONG) – output
Total number of entries available.

pcbNeeded (PULONG) – output
Size in bytes of available information.
A value of 0 specifies that the size is not known.

pReserved (PVOID) – output
Reserved.
This must be NULL.

Returns

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
ERROR_MORE_DATA (234)	Additional data is available.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_BufTooSmall (2123)	The API return buffer is too small.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

SplEnumDriver — Spooler Enumerate Driver

Remarks

The buffer contents on return are:

<i>ulLevel</i>	Buffer Contents
0	An array of PRDRIVINFO structures

Related Functions

- SplCreateDevice
- SplCreateQueue
- SplSetDevice
- SplSetQueue

Example Code

This sample code will enumerate all the drivers on a local computer.

```
#define INCL_BASE
#define INCL_DOSMEMMGR
#define INCL_SPL
#define INCL_SPLDOSPRINT

#include <os2.h>
#include <stdio.h>          /* for printf function */
#include <nterr.h>          /* for error codes */

INT main ()
{
    SPLERR splerr ;
    ULONG  cbBuf ;
    ULONG  cTotal ;
    ULONG  cReturned ;
    ULONG  cbNeeded ;
    ULONG  i ;
    PSZ    pszComputerName = NULL ;
    PSZ    pszDriverName ;
    PBYTE  pbuf ;

    /* Call the function the first time with zero in cbBuf. The count of bytes */
    /* needed for the buffer to hold all the info will be returned in cbNeeded.*/
    splerr = SplEnumDriver(pszComputerName, 0L, NULL, 0L,
                          &cReturned, &cTotal, &cbNeeded,
                          NULL );

    /* If the return code is ERROR_MORE_DATA or NERR_BufTooSmall, then
all the */
    /* parameters were correct; and we can continue. */
    if (splerr == ERROR_MORE_DATA || splerr == NERR_BufTooSmall)
    {
        /* Allocate memory for the buffer to hold the returned information. Use */
        /* the count of bytes that were returned by our first call. */
        if (!DosAllocMem( &pbuf, cbNeeded,
PAG_READ|PAG_WRITE|PAG_COMMIT ) )
        {
            /* Set count of bytes to the value returned by our first call. */
            cbBuf= cbNeeded ;

            /* Now call the function a second time with the correct values, and */
            /* the information will be returned in the buffer. */
            splerr= SplEnumDriver(pszComputerName, 0L, pbuf, cbBuf,
                                &cReturned ,&cTotal, &cbNeeded,
                                NULL ) ;

            if (splerr == NO_ERROR)
```

SplEnumDriver – Spooler Enumerate Driver

```

    {
        /* Set a pointer to point to the beginning of the buffer.          */
        pszDriverName = (PSZ)pbuf;

        /* Print the names that are in the buffer. The count of the number*/
        /* of names in pBuf have been returned in cReturned.              */
        for (i=0;i < cReturned ; i++)
        {
            printf("Driver name - %s\n", pszDriverName) ;
            /* Increment the pointer to point to the next name.            */
            pszDriverName += DRIV_NAME_SIZE +DRIV_DEVICENAME_SIZE
+ 2;
        }
    }
    /* Free the memory allocated for the buffer.                          */
    DosFreeMem(pbuf) ;
}
else
{
    /* If the first call to the function returned any error code other    */
    /* than ERROR_MORE_DATA or NERR_BufTooSmall, we print the             */
    following. */
    printf("SplEnumDriver error=%ld \n",splerr) ;
}
DosExit( EXIT_PROCESS , 0 ) ;
return (splerr);
}

```


SplEnumJob – Spooler Enumerate Job

```
#define INCL_SPL /* Or use INCL_PM */
```

```
SPLERR SplEnumJob (PSZ pszComputerName, PSZ pszQueueName, ULONG ulLevel,  
PVOID pBuf, ULONG cbBuf, PULONG pcReturned, PULONG pcTotal,  
PULONG pcbNeeded, PVOID pReserved)
```

This function lists the jobs in a print queue, optionally supplying status information on each job.

Parameters

pszComputerName (PSZ) – input
Name of computer where jobs are to be listed.
A NULL string specifies the local workstation.

pszQueueName (PSZ) – input
Queue name.

ulLevel (ULONG) – input
Level of detail required.
This must be 0 or 2.

pBuf (PVOID) – output
Buffer.

cbBuf (ULONG) – input
Size, in bytes, of Buffer.

pcReturned (PULONG) – output
Number of entries returned.

pcTotal (PULONG) – output
Number of entries available.

pcbNeeded (PULONG) – output
Size in bytes of available information.
A value of 0 specifies that the size is not known.

pReserved (PVOID) – output
Reserved. This must be NULL.

Returns

NO_ERROR (0)	No errors occurred.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
ERROR_MORE_DATA (234)	Additional data is available.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_QNotFound (2150)	The printer queue does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

SplEnumJob – Spooler Enumerate Job

Remarks

The buffer contents on return are:

<i>ulLevel</i>	Buffer Contents
0	An array containing a <i>uJobId</i> for each of <i>pcReturned</i> jobs.
2	An array containing a PRJINFO2 structure for each of <i>pcReturned</i> jobs.

Related Functions

- SplCopyJob
- SplDeleteJob
- SplQueryJob

Example Code

This sample code accepts a queue name from the command line, and then prints out all the information associated with each job in that queue. Level 0 and 2 are valid; we have chosen to print out level 2 information.

```
#define INCL_SPL
#define INCL_SPLDOSPRINT

#include <os2.h>
#include <stdio.h>      /* for printf function */
#include <neterr.h>     /* for error codes */

INT main (argc, argv)
    INT argc;
    CHAR *argv[];
{
    ULONG splerr ;
    ULONG cbBuf ;
    ULONG cTotal ;
    ULONG cReturned ;
    ULONG cbNeeded ;
    ULONG ulLevel;
    ULONG i ;
    PSZ pszComputerName ;
    PSZ pszQueueName ;
    PVOID pBuf = NULL;
    PPRJINFO2 pprj2 ;

    /* Check that the command line entry was two parameters. */
    if (argc != 2)
    {
        printf("Syntax: enumjob QueueName\n");
        DosExit( EXIT_PROCESS , 0 ) ;
    }
    /* Either a NULL or a pointer to a NULL specify the local workstation. */
    pszComputerName = (PSZ)NULL ;

    /* Set queue name equal to the value entered at the command line. */
    pszQueueName = argv[1];

    /* Valid level are 0 and 2. Level 2 gives info for a PRJINFO2 structure. */
    ulLevel = 2L;

    /* Make the call the first time with cbBuf = zero so that we can get a
    /* return of the number of bytes that are need for pBuf to hold all of
    /* the information. The bytes needed will be returned in cbNeeded. */
    splerr = SplEnumJob(pszComputerName, pszQueueName, ulLevel, pBuf, 0L,
                        &cReturned, &cTotal,
                        &cbNeeded, NULL) ;
```

SplEnumJob — Spooler Enumerate Job

```
/* Check that the return code is one of the two valid errors at this time. */
if (splerr == ERROR_MORE_DATA || splerr == NERR_BufTooSmall )
{
    /* Allocate memory for pBuf. ( No error checking is done on DosAllocMem */
    /* call to keep this sample code simple.) */
    DosAllocMem( &pBuf, cbNeeded,
                PAG_READ|PAG_WRITE|PAG_COMMIT );

    /* Set bytes needed for buffer to the value returned by the first call. */
    cbBuf = cbNeeded ;

    /* Make the call with all the valid information. */
    SplEnumJob(pszComputerName,pszQueueName, ulLevel,
                pBuf, cbBuf, &cReturned,&cTotal,
                &cbNeeded,NULL );

    /* Set up a pointer to point to the beginning of the buffer in which we */
    /* have the returned information. */
    pprj2=(PPRJINF02)pBuf;

    /* The number of structures in the buffer(pBuf) are returned in cReturned*/
    /* Implement a for loop to print out the information for each structure. */
    for (i=0; i<cReturned ;i++ )
    {
        printf("Job ID      = %d\n", pprj2->uJobId);
        printf("Job Priority = %d\n", pprj2->uPriority);
        printf("User Name   = %s\n", pprj2->pszUserName);
        printf("Position   = %d\n", pprj2->uPosition);
        printf("Status      = %d\n", pprj2->fsStatus);
        printf("Submitted  = %ld\n", pprj2->ulSubmitted);
        printf("Size        = %ld\n", pprj2->ulSize);
        printf("Comment    = %s\n", pprj2->pszComment);
        printf("Document   = %s\n\n",pprj2->pszDocument);

        /* Increment the pointer to point to the next structure in the buffer*/
        pprj2++;
    } /* endfor */
    /* Free the memory that we allocated to make the call. */
    DosFreeMem(pBuf) ;
}
else
{
    /* If any other error other than ERROR_MORE_DATA or NERR_BufTooSmall,
    then */
    /* print the returned information. */
    printf("SplEnumJob Error=%ld, Total Jobs=%ld, Returned Jobs=%ld, Bytes
    Needed=%ld\n",
           splerr, cTotal, cReturned, cbNeeded) ;
}
DosExit( EXIT_PROCESS , 0 ) ;
return (splerr);
}
```

SplEnumPort – Spooler Enumerate Port

```
#define INCL_SPL /* Or use INCL_PM */
```

```
SPLERR SplEnumPort (PSZ pszComputerName, ULONG ulLevel, PVOID pBuf, ULONG cbBuf,  
PULONG pcReturned, PULONG pcTotal, PULONG pcbNeeded,  
PVOID pReserved)
```

This function lists printer ports on the local workstation or on a remote server.

Parameters

- pszComputerName (PSZ)** – input
Name of computer where queues are to be listed.
A NULL string specifies the local workstation.
- ulLevel (ULONG)** – input
Level of detail.
The level of detail required. This must be 0 or 1.
- pBuf (PVOID)** – output
Buffer.
- cbBuf (ULONG)** – input
Size, in bytes, of Buffer.
- pcReturned (PULONG)** – output
Number of entries returned.
- pcTotal (PULONG)** – output
Total number of entries available.
- pcbNeeded (PULONG)** – output
Size in bytes of available information.
A value of 0 specifies that the size is not known.
- pReserved (PVOID)** – output
Reserved.
This must be NULL.

Returns

- | | |
|-------------------------------------|---|
| NO_ERROR (0) | No errors occurred. |
| ERROR_ACCESS_DENIED (5) | Access is denied. |
| ERROR_NOT_SUPPORTED (50) | This request is not supported by the network. |
| ERROR_BAD_NETPATH (53) | The network path cannot be located. |
| ERROR_INVALID_PARAMETER (87) | An invalid parameter is specified. |
| ERROR_INVALID_LEVEL (124) | The level parameter is invalid. |
| ERROR_MORE_DATA (234) | Additional data is available. |
| NERR_NetNotStarted (2102) | The network program is not started. |
| NERR_BufTooSmall (2123) | The API return buffer is too small. |
| NERR_SpoolerNotLoaded (2161) | The spooler is not running. |
| NERR_InvalidComputer (2351) | The computer name is invalid. |

SplEnumPort — Spooler Enumerate Port

Remarks

The buffer contents on return are:

<i>ulLevel</i>	Buffer Contents
0	An array of PRPORTINFO structures
1	An array of PRPORTINFO1 structures

Related Functions

- SplCreateDevice
- SplSetDevice

Example Code

This sample code will print out all the ports and associated information. This is done at level 1, and for the local workstation.

```
#define INCL_DOSMEMMGR
#define INCL_SPL
#define INCL_SPLDOSPRINT

#include <os2.h>
#include <stdio.h>
#include <neterr.h>

INT main ()
{
    SPLERR splerr ;
    ULONG cbBuf ;
    ULONG cTotal;
    ULONG cReturned ;
    ULONG cbNeeded ;
    ULONG ulLevel = 1;
    ULONG i ;
    PSZ pszComputerName = NULL;
    PVOID pbuf ;
    PPRPORTINFO1 pPort1 ;

    splerr = SplEnumPort(pszComputerName, ulLevel, pbuf, 0L, /* cbBuf */
                        &cReturned, &cTotal,
                        &cbNeeded, NULL) ;

    if (splerr == ERROR_MORE_DATA || NERR_BufTooSmall )
    {
        if (!DosAllocMem( &pbuf, cbNeeded,
                        PAG_READ|PAG_WRITE|PAG_COMMIT ) )
        {
            cbBuf = cbNeeded ;
            splerr = SplEnumPort(pszComputerName, ulLevel, pbuf, cbBuf,
                                &cReturned, &cTotal,
                                &cbNeeded, NULL) ;

            if (splerr == 0L)
            {
                pPort1 = (PPRPORTINFO1)pbuf ;
                printf("Port names: ");
                for (i=0; i < cReturned; i++)
                {
                    printf("Port - %s, Driver - %s Path - %s\n",
                        pPort1->pszPortName, pPort1->pszPortDriverName,
                        pPort1->pszPortDriverPathName ) ;
                    pPort1++ ;
                }
                printf("\n");
            }
        }
    }
}
```

SplEnumPort – Spooler Enumerate Port

```
        DosFreeMem(pbuf) ;
    }
else
{
    printf("SplEnumPort splerr=%ld, \n",splerr) ;
}
DosExit( EXIT_PROCESS , 0 ) ;
return (splerr);
} /* end main */
```

SplEnumPrinter – Spooler Enumerate Print Destinations

```
#define INCL_SPL /* Or use INCL_PM */
```

SPLERR SplEnumPrinter (PSZ pszComputerName, ULONG ulLevel, ULONG flType, PVOID pBuf, ULONG cbBuf, PULONG pcReturned, PULONG pcTotal, PULONG pcbNeeded, PVOID pReserved)

This function lists print destinations in the system.

Parameters

pszComputerName (PSZ) – input

Name of computer where queues are to be listed.

This must be NULL.

ulLevel (ULONG) – input

Level of detail required.

This must be 0.

flType (ULONG) – input

Type of print destinations required.

SPL_PR_QUEUE Return only queues

SPL_PR_DIRECT_DEVICE Return only direct print devices

SPL_PR_QUEUED_DEVICE Return only queued print devices

SPL_PR_LOCAL_ONLY Return only local print destinations

pBuf (PVOID) – output

Buffer.

cbBuf (ULONG) – input

Size, in bytes, of Buffer.

pcReturned (PULONG) – output

Number of entries returned.

pcTotal (PULONG) – output

Number of entries available.

pcbNeeded (PULONG) – output

Size in bytes of available information.

A value of 0 specifies that the size is not known.

pReserved (PVOID) – output

Reserved.

This must be NULL.

Returns

NO_ERROR (0)

No errors occurred.

ERROR_NOT_SUPPORTED (50)

This request is not supported by the network.

ERROR_INVALID_PARAMETER (87)

An invalid parameter is specified.

ERROR_INVALID_LEVEL (124)

The level parameter is invalid.

ERROR_MORE_DATA (234)

Additional data is available.

NERR_NetNotStarted (2102)

The network program is not started.

SplEnumPrinter – Spooler Enumerate Print Destinations

NERR_BufTooSmall (2123)

The API return buffer is too small.

NERR_SpoolerNotLoaded (2161)

The spooler is not running.

Remarks

The buffer contents on return are:

ulLevel	Buffer Contents
0	An array of PRINTERINFO structures.

When the names of print destinations are returned, calls can be made to SplQueryQueue or SplQueryDevice to find out further information about the print destination.

Related Functions

- SplQueryDevice
- SplQueryQueue

Example Code

This example code will print out all queues and printers for the local computer. It will print out both printers that are attached to a queue, and those that are direct printers.

```
#define INCL_SPL
#define INCL_SPLDOSPRINT
#include <os2.h>
#include <stdio.h>      /* for printf function */
#include <neterr.h>     /* for error codes */

INT main ()
{
    PVOID pBuf;
    ULONG fsType ;
    ULONG cbBuf ;
    ULONG cRes ;
    ULONG cTotal ;
    ULONG cbNeeded ;
    SPLERR splerr = 0 ;
    PPRINTERINFO pRes ;

    /* Set fsType to use all the flags. We will print out local device/queues.*/
    fsType = SPL_PR_QUEUE | SPL_PR_DIRECT_DEVICE |
             SPL_PR_QUEUED_DEVICE | SPL_PR_LOCAL_ONLY;

    /* Make function call with cbBuf equal to zero to get a return in cbNeeded*/
    /* of the number of bytes needed for buffer to hold all the information */

    splerr = SplEnumPrinter ( NULL,0 ,fsType ,NULL ,NULL ,&cRes ,
                             &cTotal,&cbNeeded ,NULL ) ;

    /* The error return code will be one of the two following codes if      */
    /* all the parameters were correct. Otherwise it could be                */
    /* ERROR_INVALID_PARAMETER.                                              */

    if ( splerr == ERROR_MORE_DATA || splerr == NERR_BufTooSmall )
    {
        /* Allocate memory for the buffer using the count of bytes that were */
        /* returned in cbNeeded. For simplicity, no error checking is done.   */
        DosAllocMem( &pBuf, cbNeeded,
                     PAG_READ|PAG_WRITE|PAG_COMMIT);

        /* Set count of bytes in buffer to value used to allocate buffer.    */
        cbBuf = cbNeeded;
    }
}
```


SplEnumPrinter — Spooler Enumerate Print Destinations

```

/* Call function again with the correct buffer size. */
splerr = SplEnumPrinter ( NULL,0 ,fsType ,pBuf ,cbBuf ,&cRes ,
                        &cTotal,&cbNeeded,NULL);

/* If there are any returned structures in the buffer, then we will */
/* print out some of the information. */
if (cRes)
{
    pRes = (PPRINTERINFO)pBuf ;
    while ( cRes-- )
    {
        /* Look at the flType element in the pRes structure to determine */
        /* what type of print destination the structure represents. */
        switch (pRes[cRes].flType)
        {
            case SPL_PR_QUEUE:
                printf("Print destination %s is a queue.\n",
                    pRes[cRes].pszPrintDestinationName) ;
                break;
            case SPL_PR_QUEUED_DEVICE:
                printf("Print destination %s is a queued printer.\n",
                    pRes[cRes].pszPrintDestinationName) ;
                break;
            case SPL_PR_DIRECT_DEVICE:
                printf("Print destination %s is a direct printer.\n",
                    pRes[cRes].pszPrintDestinationName) ;
        }
        printf("Description -
            %s\n\n",pRes[cRes].pszDescription) ;
    }
}
DosFreeMem(pBuf);
}
else
{
    /* If we had any other return code other than ERROR_MORE_DATA or */
    /* NERR_BufTooSmall, we will print out the following information. */
    printf("SplEnumPrinter error= %ld \n",splerr);
}
DosExit( EXIT_PROCESS , 0 ) ;
return (splerr);
}

```

SplEnumQueue – Spooler Enumerate Queue

```
#define INCL_SPL /* Or use INCL_PM */
```

**SPLERR SplEnumQueue (PSZ pszComputerName, ULONG ulLevel, PVOID pBuf,
ULONG cbBuf, PULONG pcReturned, PULONG pcTotal,
PULONG pcbNeeded, PVOID pReserved)**

This function lists print queues on the local workstation or on a remote server, optionally supplying additional information.

Parameters

- pszComputerName (PSZ)** – input
Name of computer where queues are to be listed.
A NULL string specifies the local workstation.
- ulLevel (ULONG)** – input
Level of detail.
The level of detail required. This must be 3, 4, 5 or 6.
- pBuf (PVOID)** – output
Buffer.
- cbBuf (ULONG)** – input
Size, in bytes, of Buffer.
- pcReturned (PULONG)** – output
Number of entries returned.
- pcTotal (PULONG)** – output
Total number of entries available.
- pcbNeeded (PULONG)** – output
Size in bytes of available information.
A value of 0 specifies that the size is not known.
- pReserved (PVOID)** – output
Reserved.
This must be NULL.

Returns

Return

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
ERROR_MORE_DATA (234)	Additional data is available.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_BufTooSmall (2123)	The API return buffer is too small.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.

SplEnumQueue — Spooler Enumerate Queue

NERR_InvalidComputer (2351)

The computer name is invalid.

Remarks

The buffer contents on return are:

<i>ulLevel</i>	Buffer Contents
3	An array of PRQINFO3 structures
4	An array of <i>pcReturned</i> PRQINFO3 structures in which each PRQINFO3 structure is followed by an array of PRJINFO2 structures, one for each of job in the queue. <i>cJobs</i> in the PRQINFO3 or PRQINFO6 structure gives the number of jobs in the array.
5	A queue name of type PSZ.
6	An array of PRQINFO6 structures

Related Functions

- SplQueryJob
- SplQueryQueue
- SplSetJob
- SplSetQueue

Example Code

This sample code enumerates all the queues and the jobs in them that are on the local workstation.

```
#define INCL_BASE
#define INCL_SPL
#define INCL_SPLDOSPRINT
#include <os2.h>
#include <stdio.h>
#include <neterr.h>

INT main ()
{
    SPLERR splerr;
    USHORT jobCount ;
    ULONG cbBuf ;
    ULONG cTotal;
    ULONG cReturned ;
    ULONG cbNeeded ;
    ULONG ulLevel ;
    ULONG i,j ;
    PSZ pszComputerName ;
    PBYTE pBuf ;
    PPRQINFO3 prq ;
    PPRJINFO2 prj2 ;

    ulLevel = 4L;
    pszComputerName = (PSZ)NULL ;
    splerr = SplEnumQueue(pszComputerName, ulLevel, pBuf, 0L, /* cbBuf */
                        &cReturned, &cTotal,
                        &cbNeeded, NULL)
    if ( splerr == ERROR_MORE_DATA || splerr == NERR_BufTooSmall )
    {
        if (!DosAllocMem( &pBuf, cbNeeded,
                        PAG_READ|PAG_WRITE|PAG_COMMIT)
    )
    {
        cbBuf = cbNeeded ;
        splerr = SplEnumQueue(pszComputerName, ulLevel, pBuf, cbBuf,
                        &cReturned, &cTotal,
                        &cbNeeded, NULL)
        if (splerr == NO_ERROR)
        {

```

SplEnumQueue – Spooler Enumerate Queue

```

/* Set pointer to point to the beginning of the buffer.          */
prq = (PPRQINF03)pBuf ;

/* cReturned has the count of the number of PRQINF03 structures. */
for (i=0; i < cReturned ; i++)
{
    printf("Queue info: name - %s\n", prq->pszName) ;
    printf(" priority - %d starttime - %d endtime - %d fsType -
%X\n",
        prq->uPriority , prq->uStartTime , prq->uUntilTime ,
        prq->fsType ) ;
    printf(" pszSepFile - %s\n", prq->pszSepFile) ;
    printf(" pszPrProc - %s\n", prq->pszPrProc) ;
    printf(" pszParms - %s\n", prq->pszParms) ;
    printf(" pszComment - %s\n", prq->pszComment) ;
    printf(" pszPrinters - %s\n", prq->pszPrinters) ;
    printf(" pszDriverName- %s\n", prq->pszDriverName) ;
    if (prq->pDriverData)
    {
        printf(" pDriverData->cb - %ld\n",
(ULONG)prq->pDriverData->cb);
        printf(" pDriverData->lVersion - %ld\n",
(ULONG)prq->pDriverData->lVersion) ;
        printf(" pDriverData->szDeviceName- %s\n",
prq->pDriverData->szDeviceName) ;
    }
    /* Save the count of jobs. There are this many PRJINFO2      */
    /* structures following the PRQINF03 structure.              */
    jobCount = prq->cJobs;
    printf("Job count in this queue is %d\n\n", jobCount);

    /* Increment the pointer past the PRQINF03 structure.        */
    prq++;

    /* Set a pointer to point to the first PRJINFO2 structure.   */
    prj2=(PPRJINFO2)prq;
    for (j=0; j < jobCount ; j++)
    {
        printf("Job ID = %d\n", prj2->uJobId);
        printf("Job Priority= %d\n", prj2->uPriority);
        printf("User Name = %s\n", prj2->pszUserName);
        printf("Position = %d\n", prj2->uPosition);
        printf("Status = %d\n", prj2->fsStatus);
        printf("Submitted = %ld\n", prj2->ulSubmitted);
        printf("Size = %ld\n", prj2->ulSize);
        printf("Comment = %s\n", prj2->pszComment);
        printf("Document = %s\n\n", prj2->pszDocument);

        /* Increment the pointer to point to the next structure. */
        prj2++;
    } /* endfor jobCount */

    /* After doing all the job structures, prj2 points to the next */
    /* queue structure. Set the pointer for a PRQINF03 structure. */
    prq = (PPRQINF03)prj2;
} /*endfor cReturned */
}
DosFreeMem(pBuf) ;
} /* end if Q level given */
else
{
    /* If we are here we had a bad error code. Print it and some other info.*/
    printf("SplEnumQueue Error=%ld, Total=%ld, Returned=%ld, Needed=%ld\n",

```

SplEnumQueue — Spooler Enumerate Queue

```
        splerr, cTotal, cReturned, cbNeeded) ;  
    }  
    DosExit( EXIT_PROCESS , 0 ) ;  
    return(splerr);  
} /* end main */
```

SpiEnumQueueProcessor – Spooler Enumerate Queue Processor

```
#define INCL_SPL /* Or use INCL_PM */
```

SPLERR SpiEnumQueueProcessor (PSZ pszComputerName, ULONG ulLevel, PVOID pBuf, ULONG cbBuf, PULONG pcReturned, PULONG pcTotal, PULONG pcbNeeded, PVOID pReserved)

This function lists printer queue processors on the local workstation or on a remote server.

Parameters

pszComputerName (PSZ) – input
Name of computer where queues are to be listed.
A NULL string specifies the local workstation.

ulLevel (ULONG) – input
Level of detail.
The level of detail required. This must be 0.

pBuf (PVOID) – output
Buffer.

cbBuf (ULONG) – input
Size, in bytes, of Buffer.

pcReturned (PULONG) – output
Number of entries returned.

pcTotal (PULONG) – output
Total number of entries available.

pcbNeeded (PULONG) – output
Size in bytes of available information.
A value of 0 specifies that the size is not known.

pReserved (PVOID) – output
Reserved.
This must be NULL.

Returns

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
ERROR_MORE_DATA (234)	Additional data is available.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_BufTooSmall (2123)	The API return buffer is too small.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

SplEnumQueueProcessor — Spooler Enumerate Queue Processor

Remarks

The buffer contents on return are:

<i>ulLevel</i>	Buffer Contents
0	An array of PRQPROCINFO structures

Related Functions

- SplSetQueue

Example Code

This sample code enumerates and prints all the queue processors on the local computer.

```
#define INCL_BASE
#define INCL_SPL
#define INCL_SPLDOSPRINT

#include <os2.h>
#include <stdio.h>          /* for printf function */
#include <neterr.h>         /* for error codes */

INT main ()
{
    SPLERR splerr ;
    ULONG  cbBuf ;
    ULONG  cTotal ;
    ULONG  cReturned ;
    ULONG  cbNeeded ;
    ULONG  i ;
    PSZ    pszComputerName = NULL ;
    PSZ    pszQProcName ;
    PBYTE  pBuffer ;

    /* Call the function the first time with zero in cbBuf. The count */
    /* of bytes needed for the buffer to hold all the info will be */
    /* returned in cbNeeded. */
    splerr = SplEnumQueueProcessor(pszComputerName, 0L, NULL, 0L,
                                   &cReturned, &cTotal,
                                   &cbNeeded, NULL );

    /* If the return code is ERROR_MORE_DATA or NERR_BufTooSmall, */
    /* then all the parameters were correct; and we can continue. */
    if (splerr == ERROR_MORE_DATA || splerr == NERR_BufTooSmall)
    {
        /* Allocate memory for the buffer to hold the returned information. Use */
        /* the count of bytes that were returned by our first call. */
        if (!DosAllocMem( &pBuffer, cbNeeded,
                          PAG_READ|PAG_WRITE|PAG_COMMIT ) )
        {
            /* Set count of bytes to the value returned by our first call. */
            cbBuf = cbNeeded ;

            /* Now call the function a second time with the correct values, and */
            /* the information will be returned in the buffer. */
            splerr = SplEnumQueueProcessor(pszComputerName, 0L, pBuffer, cbBuf,
                                           &cReturned, &cTotal,
                                           &cbNeeded, NULL );

            /* If we have no errors, then print out the buffer information. */
            if (splerr == NO_ERROR)
            {
                /* Set a pointer to point to the beginning of the buffer. */
                pszQProcName = (PSZ)pBuffer;
            }
        }
    }
}
```

SplEnumQueueProcessor — Spooler Enumerate Queue Processor

```
/* Print the names that are in the buffer. The count of the number*/
/* of names in pBuf have been returned in cReturned. */
for (i=0;i < cReturned ; i++)
{
    printf("Queue Processor name - %s\n", pszQProcName) ;

    /* Increment the pointer to point to the next name. */
    pszQProcName += DRIV_NAME_SIZE + 1;
}
/* Free the memory allocated for the buffer. */
DosFreeMem(pBuf) ;
}
else
{
    /* If the first call to the function returned any other error code */
    /* except ERROR_MORE_DATA or NERR_BufTooSmall, we print the */
    /* following. */
    printf("SplEnumQueueProcessor error=%ld\n",splerr) ;
}
DosExit( EXIT_PROCESS , 0 ) ;
return (splerr);
}
```


SplHoldJob — Spooler Hold Job

```
#define INCL_SPL /* Or use INCL_PM */
```

SPLERR SplHoldJob (PSZ pszComputerName, PSZ pszQueueName, ULONG ulJob)

This function holds a job in a print queue.

Parameters

pszComputerName (PSZ) — input

Name of computer where job is to be paused.

A NULL string specifies the local workstation.

pszQueueName (PSZ) — input

Queue Name.

ulJob (ULONG) — input

Job identification number.

Returns

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
NERR_NetNotStarted (2102)	The network program is not installed.
NERR_JobNotFound (2151)	The print job does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_JobInvalidState (2164)	This operation cannot be performed on the print job in its current state.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

If the job is already printing when the call is made, NERR_JobInvalidState (2164) is returned.

A user with administrator privilege can hold any job.

A job created locally can be held locally regardless of user privilege level, but can be held remotely only by an administrator.

A remote job can be held by a user without administrator privilege only if the username of the person initiating the request is the same as the username of the person who created the job.

Related Functions

- SplEnumJob
- SplQueryJob
- SplReleaseJob

Example Code

This sample code will hold the queue name that is entered at the prompt.

```
#define INCL_BASE
#define INCL_SPL
#include <os2.h>
#include <stdio.h>      /* for printf function */
#include <neterr.h>     /* for error codes   */
#include <stdlib.h>     /* for atoi function */

INT main (argc, argv)
    INT argc;
    CHAR *argv[];
{
    SPLERR splerr ;
    PSZ    pszComputerName = NULL ;
    PSZ    pszQueueName = NULL ;
    ULONG  ulJob ;

    /* Get job id from the input argument. */
    ulJob = atoi(argv[1]);

    /* Call the function to do the hold. If an error is returned, print it. */
    splerr = SplHoldJob( pszComputerName, pszQueueName, ulJob);

    switch (splerr)
    {
        case NO_ERROR:
            printf("Job %d was held.\n",ulJob);
            break;
        case NERR_JobNotFound:
            printf("Job does not exist.\n");
            break;
        case NERR_JobInvalidState:
            printf("This operation can't be performed on the print Job.\n");
            break;
        default:
            printf("Errorcode = %ld\n",splerr);
    } /* endswitch */
    DosExit( EXIT_PROCESS , 0 ) ;
    argc;
    return (splerr);
}
```

SplHoldQueue — Spooler Hold Queue

```
#define INCL_SPL /* Or use INCL_PM */
```

SPLERR SplHoldQueue (PSZ pszComputerName, PSZ pszQueueName)

This function holds a print queue.

Parameters

pszComputerName (PSZ) — input
Name of computer where queue is to be paused.
A NULL string specifies the local workstation.

pszQueueName (PSZ) — input
Queue name.

Returns

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_QNotFound (2150)	The printer queue does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

This function suspends processing of all print jobs except for a job currently printing. Print jobs can be submitted to a held queue, but no jobs will be spooled to a print destination or print processor until the queue is released by a SplHoldQueue call.

To hold a remote queue requires administrator privilege on the remote server.

Related Functions

- SplCreateQueue
- SplEnumQueue
- SplQueryQueue
- SplReleaseQueue

SplHoldQueue – Spooler Hold Queue

Example Code

This sample code will hold the local queue name that is entered at the prompt.

```
#define INCL_SPL
#include <os2.h>
#include <stdio.h>      /* for printf function */
#include <neterr.h>     /* for error codes */

INT main (argc, argv)
    INT argc;
    CHAR *argv[];
{
    SPLERR splerr ;
    PSZ    pszComputerName = NULL ;
    PSZ    pszQueueName ;

    /* Get queue name from the input argument */
    pszQueueName = argv[1];

    /* Call the function to do the hold. If an error is returned, print it. */
    splerr = SplHoldQueue(pszComputerName, pszQueueName);
    if (splerr != 0L)
    {
        switch (splerr)
        {
            case NERR_QNotFound:
                printf("Queue does not exist.\n");
                break;
            case NERR_SpoolerNotLoaded:
                printf("The Spooler is not running.\n");
                break;
            default:
                printf("Errorcode = %ld\n", splerr);
        } /* endswitch */
    }
    else
    {
        printf("Queue %s was held.\n", pszQueueName);
    } /* endif */
    DosExit( EXIT_PROCESS , 0 ) ;
    argc; /* keep the compiler quiet */
    return (splerr);
}
```

SplPurgeQueue – Spooler Purge Queue

```
#define INCL_SPL /* Or use INCL_PM */
```

SPLERR SplPurgeQueue (PSZ pszComputerName, PSZ pszQueueName)

This function removes all jobs, except any currently printing, from a print queue.

Parameters

pszComputerName (PSZ) – input

Name of computer where queue is to be purged.

A NULL string specifies the local workstation.

pszQueueName (PSZ) – input

Queue name.

Returns

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter was specified.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_QNotFound (2150)	The printer queue does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

A print job that is printing is not affected by this function.

If a print queue is pending deletion when this function is made, the queue is deleted when the job that is currently printing ends.

To purge a remote queue requires administrator privilege on the remote server.

Related Functions

- SplCreateQueue
- SplEnumQueue
- SplQueryQueue

SplPurgeQueue — Spooler Purge Queue

Example Code

This code will purge a local queue, whose name is entered at the prompt.

```
#define INCL_SPL
#include <os2.h>
#include <stdio.h>      /* for printf function */
#include <neterr.h>     /* for error codes */

INT main (argc, argv)
    INT argc;
    CHAR *arg[];
{
    SPLERR splerr ;
    PSZ   pszComputerName = NULL ;
    PSZ   pszQueueName ;

    /* Get queue name from the input argument. */
    pszQueueName = arg[1];

    /* Call the function to do the purge. If an error is returned, print it. */
    splerr=SplPurgeQueue(pszComputerName, pszQueueName);
    if (splerr != 0L)
    {
        switch (splerr)
        {
            case NERR_QNotFound:
                printf("Queue does not exist.\n");
                break;
            case NERR_SpoolerNotLoaded:
                printf("The Spooler is not running.\n");
                break;
            default:
                printf("Errorcode = %ld\n",splerr);
        } /* endswitch */
    }
    else
    {
        printf("Queue %s was purged.\n",pszQueueName);
    } /* endif */

    DosExit( EXIT_PROCESS , 0 ) ;
    return (splerr);
}
```

SplQmAbort – Spooler File Abort

```
#define INCL_SPL /* Or use INCL_PM */
```

BOOL SplQmAbort (HSPL hspl)

This function stops the generation of the spool file(s). It automatically closes the spool file (see SplQmClose).

Parameters

hspl (HSPL) – input
Spooler handle.

Returns

Success indicator:

TRUE	Successful completion
FALSE	Error occurred.

Possible returns from WinGetLastError

PMERR_SPL_QUEUE_ERROR	No spooler queue supplied or found.
PMERR_SPL_INV_HSPL	The spooler handle is invalid.

Related Functions

Prerequisite Functions

- SplQmOpen

Other Related Functions

- DevEscape

Example Code

This function is used to stop the generation of spool files and automatically close the spool file.

```
#define INCL_SPL
#include <OS2.H>

HSPL hspl; /* spooler handle. */

SplQmAbort(hspl);
```

SpiQmAbortDoc – Spooler File Abort Document

```
#define INCL_SPL /* Or use INCL_PM */
```

BOOL SpiQmAbortDoc (HSPL hspl)

This function aborts a print job.

Parameters

hspl (HSPL) – input
Spooler handle.

Returns

Success indicator:

TRUE	Successful completion
FALSE	Error occurred.

Possible returns from WinGetLastError

PMERR_SPL_QUEUE_ERROR

No spooler queue supplied or found.

PMERR_STARTDOC_NOT_ISSUED

A request to write spooled output without first issuing a STARTDOC was attempted.

PMERR_SPL_INV_HSPL

The spooler handle is invalid.

Remarks

Everything that has been written to the spool file for this job since the last SpiQmStartDoc is erased, including the SpiQmStartDoc.

Related Functions

Prerequisite Functions

- SpiQmOpen
- SpiQmStartDoc

Other Related Functions

- DevEscape

Example Code

This function is used to abort a print job. Everything since the last SpiQmStartDoc is deleted.

```
#define INCL_SPL  
#include <OS2.H>
```

```
HSPL hspl; /* spooler handle. */
```

```
SpiQmAbortDoc(hspl);
```


SplQmClose – Spool File Close

```
#define INCL_SPL /* Or use INCL_PM */
```

BOOL SplQmClose (HSPL hspl)

This function corresponds to the DevCloseDC function: it closes the spool file.

Parameters

hspl (HSPL) – input
Spooler handle.

Returns

Success indicator:

TRUE	Successful completion
FALSE	Error occurred.

Possible returns from WinGetLastError

PMERR_SPL_QUEUE_ERROR

No spooler queue supplied or found.

PMERR_ENDDOC_NOT_ISSUED

A request to close the spooled output without first issuing an ENDDOC was attempted.

PMERR_SPL_INV_HSPL

The spooler handle is invalid.

Related Functions

Prerequisite Functions

- SplQmOpen

Other Related Functions

- DevCloseDC

Example Code

This function is used to close a spool file that was opened with SplQmOpen.

```
#define INCL_SPL  
#include <OS2.H>
```

```
HSPL hspl; /* spooler handle. */
```

```
SplQmClose(hspl);
```

SplQmEndDoc – Spooler File End Document

```
#define INCL_SPL /* Or use INCL_PM */
```

ULONG SplQmEndDoc (HSPL hspl)

This function corresponds to the DevEscape (DEVESC_ENDDOC) call: it ends a print job, and returns *ulJob*, a unique number to identify the job.

Parameters

hspl (HSPL) – input
Spooler handle.

Returns

Job identifier:

Nonzero Jobid (1 through 65 535)

SPL_ERROR Error.

Possible returns from WinGetLastError

PMERR_SPL_QUEUE_ERROR No spooler queue supplied or found.

PMERR_SPL_NO_DATA No data supplied or found.

PMERR_SPL_INV_HSPL The spooler handle is invalid.

Remarks

The print-job identifier is displayed to the user by the spooler while this job is on the queue, and while it is being printed.

Related Functions

Prerequisite Functions

- SplQmOpen
- SplQmStartDoc

Other Related Functions

- DevEscape

SplQmEndDoc — Spooler File End Document

Example Code

This function is used to end a print job and return the job id.

```
#define INCL_SPL
#include <OS2.H>

HSPL hspl; /* spooler handle. */
ULONG jobid;
CHAR szMsg[100];
HWND hwndClient;

jobid = SplQmEndDoc(hspl);

sprintf(szMsg, "ending job %d", jobid);
WinMessageBox(HWND_DESKTOP,
    hwndClient,          /* client-window handle */
    szMsg,               /* body of the message */
    "Printing Information", /* title of the message */
    0,                  /* message box id */
    MB_NOICON | MB_OK); /* icon and button flags */
```

SplQmOpen – Spooler File Open

```
#define INCL_SPL /* Or use INCL_PM */
```

HSPL SplQmOpen (PSZ pszToken, LONG ICount, PQMOPENDATA pqmdopData)

This function corresponds to the DevOpenDC call: it opens a spool file for generating a print job.

Parameters

pszToken (PSZ) – input

A token (nickname) that identifies spooler information.

This information is held in the initialization file, and is the same as that in *pqmdopData*; any that is obtained from *pqmdopData* overrides the information obtained using *pszToken*.

If *pszToken* is specified as "*", then no device information is taken from the initialization file.

Presentation Manager behaves as if "*" is specified, but it allows any string to be specified.

ICount (LONG) – input

Number of items.

This is the number of items present in the *pqmdopData* supplied. This can be shorter than the full list, if omitted items are irrelevant, or supplied from *pszToken* or elsewhere.

pqmdopData (PQMOPENDATA) – input

Open parameters.

Returns

Spooler handle:

Nonzero Spooler handle

SPL_ERROR Error.

Possible returns from WinGetLastError

PMERR_INVALID_PARM

A parameter to the function contained invalid data.

PMERR_SPL_INV_LENGTH_OR_COUNT

The length or count is invalid.

PMERR_SPL_QUEUE_NOT_FOUND

The spooler queue definition could not be found.

PMERR_BASE_ERROR

An OS/2 base error has occurred. The base error code can be accessed using the OffBinaryData field of the ERRINFO structure returned by WinGetErrorInfo.

Remarks

None

Related Functions

- DevOpenDC

SplQmOpen — Spooler File Open

Example Code

This sample code will initialize a PDEVOPENSTRUC and use it to call the function.

```
#define INCL_SPL
#define INCL_SPLDOSPRINT
#define INCL_BASE
#define INCL_ERRORS

#include <os2.h>
#include <stdio.h>
#include <stdlib.h>

VOID main()
{
    HSPL hspl;
    PDEVOPENSTRUC pdata;          /* Pointer to a DEVOPENSTRUC structure */
    PSZ pszToken = "";           /* Spooler info identifier */

    /* Allocate memory for pdata */
    if ( !DosAllocMem( &pdata, sizeof( DEVOPENSTRUC ),
        (PAG_READ|PAG_WRITE|PAG_COMMIT ) )
    {
        /* Initialize elements of pdata */
        pdata->pszLogAddress      = "LPT1Q1";
        pdata->pszDriverName      = "IBMNULL";
        pdata->pdriv               = NULL;
        pdata->pszDataType         = "PM_Q_STD";
        pdata->pszComment          = NULL;
        pdata->pszQueueProcName    = NULL;
        pdata->pszQueueProcParams  = NULL;
        pdata->pszSpoolerParams    = NULL;
        pdata->pszNetworkParams    = NULL;

        hspl = SplQmOpen( pszToken, 4L, ( PQMOPENDATA )pdata );

        if ( hspl != SPL_ERROR )    /* Good spooler handle */
        {
            printf("SplQmOpen handle is %d\n", hspl);
        }
        else
        {
            printf("SplQmOpen failed.\n");
        }
    }
}
```

SplQmStartDoc – Spooler File Start Document

```
#define INCL_SPL /* Or use INCL_PM */
```

BOOL SplQmStartDoc (HSPL hspl, PSZ pszDocName)

This function corresponds to the DevEscape (DEVESC_STARTDOC) call; it starts a print job.

Parameters

hspl (HSPL) – input
Spooler handle.

pszDocName (PSZ) – input
Document name.

This is part of the job description which is displayed to the end user by the spooler.

Returns

Success indicator:

TRUE	Successful completion
FALSE	Error occurred.

Possible returns from WinGetLastError

PMERR_INVALID_PARM	A parameter to the function contained invalid data.
PMERR_SPL_QUEUE_ERROR	No spooler queue supplied or found.
PMERR_ENDDOC_NOT_ISSUED	A request to close the spooled output without first issuing an ENDDOC was attempted.
PMERR_SPL_INV_HSPL	The spooler handle is invalid.

Remarks

This function signifies the start of a print job. It allows the application to specify a document name to be associated with the print job.

Multiple print jobs can be generated, within a single queue manager open, by bracketing each job with SplQmStartDoc and SplQmEndDoc.

Related Functions

Prerequisite Functions

- SplQmOpen

Other Related Functions

- DevEscape

SplQmStartDoc – Spooler File Start Document

Example Code

This function is used to start a print job.

```
#define INCL_SPL
#include <OS2.H>

HSPL hspl; /* spooler handle. */
CHAR szDocName[] = "Test Job";
CHAR szMsg[100];
HWND hwndClient;

sprintf(szMsg, "Starting job named: %s", szDocName);
WinMessageBox(HWND_DESKTOP,
    hwndClient,          /* client-window handle */
    szMsg,               /* body of the message */
    "Printing Information", /* title of the message */
    0,                  /* message box id */
    MB_NOICON | MB_OK); /* icon and button flags */

SplQmStartDoc(hspl, szDocName);
```

SplQmWrite – Spooler File Write

```
#define INCL_SPL /* Or use INCL_PM */
```

BOOL SplQmWrite (HSPL hspl, LONG ICount, PVOID pData)

This function writes a buffer of data to the spool file for the print job.

Parameters

hspl (HSPL) – input
Spooler handle.

ICount (LONG) – input
Length in bytes.

This is the length of *pData*; it must not be greater than 65 535. Data that is longer than this must be written by two or more calls.

pData (PVOID) – input
Buffer of data to be written to the spool file.

Returns

Success indicator:

TRUE Successful completion

FALSE Error occurred.

Possible returns from WinGetLastError

PMERR_INVALID_PARM

A parameter to the function contained invalid data.

PMERR_BASE_ERROR

An OS/2 base error has occurred. The base error code can be accessed using the OffBinaryData field of the ERRINFO structure returned by WinGetErrorInfo.

PMERR_SPL_INV_LENGTH_OR_COUNT

The length or count is invalid.

PMERR_SPL_QUEUE_ERROR

No spooler queue supplied or found.

PMERR_SPL_PRINT_ABORT

The job has already been aborted.

PMERR_STARTDOC_NOT_ISSUED

A request to write spooled output without first issuing a STARTDOC was attempted.

PMERR_SPL_CANNOT_OPEN_FILE

Unable to open the file.

PMERR_SPL_INV_HSPL

The spooler handle is invalid.

PMERR_SPL_NO_DISK_SPACE

There is not enough free disk space.

Remarks

None

Related Functions

Prerequisite Functions

- SplQmOpen
- SplQmStartDoc

SplQmWrite — Spooler File Write

Example Code

This function writes a buffer of data to the spool file for the print job.

```
#define INCL_SPL
#include <OS2.H>

HSPL hsp1; /* spooler handle. */

SplQmWrite(hsp1,
            sizeof("DATA"),
            (PVOID)"DATA");
```

SplQueryDevice – Spooler Query Device

```
#define INCL_SPL /* Or use INCL_PM */
```

SPLERR SplQueryDevice (PSZ pszComputerName, PSZ pszPrintDeviceName, ULONG ulLevel, PVOID pBuf, ULONG cbBuf, PULONG pcbNeeded)

This function retrieves information about a print device.

Parameters

pszComputerName (PSZ) – input

Name of computer where print device is to be queried.

A NULL string specifies the local workstation.

pszPrintDeviceName (PSZ) – input

Name of Print Device.

This can specify a print device name or a port name. If *ulLevel* is 0, it must be a port name. If *ulLevel* is 2 or 3, it must be a print device name.

ulLevel (ULONG) – input

Level of detail required.

This must be 0, 2 or 3.

pBuf (PVOID) – output

Buffer.

cbBuf (ULONG) – input

Size, in bytes, of Buffer.

pcbNeeded (PULONG) – output

Size in bytes of available information.

Returns

NO_ERROR (0)	No errors occurred.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
ERROR_MORE_DATA (234)	Additional data is available.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_BufTooSmall (2123)	The API return buffer is too small.
NERR_DestNotFound (2152)	The print device cannot be found.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

SplQueryDevice — Spooler Query Device

Remarks

The buffer contents on return are:

<i>ulLevel</i>	Buffer Contents
0	A port name.
2	A print device name. of type PSZ.
3	A PRDINFO3 structure.

If *ulLevel* is 3, and *pBuf* cannot hold the entire PRDINFO3 structure, *SplQueryDevice* returns *NERR_BufTooSmall* (2123).

To obtain the size of buffer required, call *SplQueryDevice* with the required value of *ulLevel* and a NULL buffer. The number of bytes required is returned in *pcbNeeded*.

If no job is printing on the print device, bits 2–11 of *fsStatus* in the PRDINFO3 data structure are meaningless.

Related Functions

- *SplCreateDevice*
- *SplDeleteDevice*
- *SplEnumDevice*

Example Code

This sample code returns information for the device name that is entered at the command line. The local workstation is selected. The query is done for level 3 information.

```
#define INCL_BASE
#define INCL_DOSMEMMGR
#define INCL_SPL
#define INCL_SPLDOSPRINT

#include <os2.h>
#include <stdio.h>
#include <neterr.h>

INT main (argc, argv)
    INT argc;
    CHAR *argv[];
{
    SPLERR splerr ;
    ULONG cbBuf;
    ULONG cbNeeded ;
    ULONG ulLevel ;
    PSZ pszComputerName ;
    PSZ pszPrintDeviceName ;
    PVOID pBuf ;
    PPRDINFO3 pprd3 ;

    if (argc != 2)
    {
        printf("Syntax: sdqry DeviceName \n");
        DosExit( EXIT_PROCESS , 0 ) ;
    }

    pszComputerName = (PSZ)NULL ;
    pszPrintDeviceName = argv[1];
    ulLevel = 3;
    splerr = SplQueryDevice(pszComputerName, pszPrintDeviceName,
                           ulLevel, (PVOID)NULL, 0L, &cbNeeded );
    if (splerr != NERR_BufTooSmall)
    {
```

SplQueryDevice – Spooler Query Device

```
    printf("SplQueryDevice Err=%ld, cbNeeded=%ld\n", splerr, cbNeeded) ;
    DosExit( EXIT_PROCESS , 0 ) ;
}
if (!DosAllocMem( &pBuf, cbNeeded,
                  PAG_READ|PAG_WRITE|PAG_COMMIT) ){
    cbBuf= cbNeeded ;
    splerr = SplQueryDevice(pszComputerName, pszPrintDeviceName,
                           ulLevel, pBuf, cbBuf, &cbNeeded) ;

    printf("SplQueryDevice Error=%ld, Bytes Needed=%ld\n", splerr,
           cbNeeded) ;

    pprd3=(PPRDINFO3)pBuf;

    printf("Print Device info: name - %s\n", pprd3->pszPrinterName) ;
    printf("User Name      = %s\n", pprd3->pszUserName) ;
    printf("Logical Address= %s\n", pprd3->pszLogAddr) ;
    printf("Job ID        = %d\n", pprd3->uJobId) ;
    printf("Status         = %d\n", pprd3->fsStatus) ;
    printf("Status Comment = %s\n", pprd3->pszStatus) ;
    printf("Comment        = %s\n", pprd3->pszComment) ;
    printf("Drivers         = %s\n", pprd3->pszDrivers) ;
    printf("Time           = %d\n", pprd3->time) ;
    printf("Time Out        = %d\n", pprd3->usTimeOut) ;
    DosFreeMem(pBuf) ;
}
DosExit( EXIT_PROCESS , 0 ) ;
return (splerr);
}
```

SplQueryJob – Spooler Query Job

```
#define INCL_SPL /* Or use INCL_PM */
```

**SPLERR SplQueryJob (PSZ pszComputerName, PSZ pszQueueName, ULONG ulJob,
ULONG ulLevel, PVOID pBuf, ULONG cbBuf, PULONG pcbNeeded)**

This function retrieves information about a print job.

Parameters

pszComputerName (PSZ) – input
Name of computer where print job is to be queried.
A NULL string specifies the local workstation.

pszQueueName (PSZ) – input
Queue Name.

ulJob (ULONG) – input
Job identification number.

ulLevel (ULONG) – input
Level of detail required.
This must be 0, 2, or 3.

pBuf (PVOID) – output
Buffer.

cbBuf (ULONG) – input
Size, in bytes, of Buffer.

pcbNeeded (PULONG) – output
Size in bytes of available information.

Returns

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
ERROR_MORE_DATA (234)	Additional data is available.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_BufTooSmall (2123)	The API return buffer is too small.
NERR_JobNotFound (2151)	The print job does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

The buffer contents on return are:

<i>ulLevel</i>	Buffer Contents
0	The job identifier
2	A PRJINFO2 structure, with variable information, up to the <i>cbBuf</i> of <i>pBuf</i>
3	A PRJINFO3 structure, with variable information, up to the <i>cbBuf</i> of <i>pBuf</i> .

Related Functions

- SplEnumJob
- SplEnumQueue
- SplQueryQueue
- SplSetJob

Example Code

The following sample code will print out the information contained in a PRJINFO3 structure that is returned from a SplQueryJob call. The parameters that are entered on the command line are the computer name, queue name, and the job id.

```
#define INCL_SPL
#define INCL_SPLDOSPRINT
#include <os2.h>
#include <stdio.h> /* for printf call */
#include <stdlib.h> /* for atoi call */
#include <neterr.h> /* for error codes */

INT main (argc, argv)
    INT argc;
    CHAR *argv[];
{
    INT    splerr;
    ULONG  cbBuf ;
    ULONG  cbNeeded ;
    ULONG  ulLevel ;
    ULONG  ulJob ;
    PSZ    pszComputerName ;
    PSZ    pszQueueName ;
    PVOID  pBuf;
    PPRJINFO3 pprj3 ;

    /* Input the parameters Computer Name, Queue Name, and Job ID. Check that
    /* three parameters have been entered along with the program name.
    if (argc != 4)
    {
        /* Print a message and exit if wrong number of parameters entered
        printf("Syntax: sjqry ComputerName QueueName JobID \n");
        DosExit( EXIT_PROCESS , 0 ) ;
    }
    /* Set the parameters to the values entered on the command line.
    pszComputerName = argv[1] ;
    pszQueueName = argv[2] ;
    ulJob = atoi (argv[3]);

    /* Valid levels are 0,2,and 3. Level 3 returns a PRJINFO3 structure.
    ulLevel = 3 ;

    /* Call the function with cbBuf equal to zero in order to get the number
    /* of bytes needed returned in cbNeeded.
    splerr = SplQueryJob(pszComputerName,pszQueueName,ulJob,
                        ulLevel, (PVOID)NULL, 0L, &cbNeeded );
```

SplQueryJob – Spooler Query Job

```

/* Only continue if the error return code is one of the two following. */
if (splerr == NERR_BufTooSmall || splerr == ERROR_MORE_DATA )
{
    /* Allocate memory for the buffer(pBuf). Only continue if memory is */
    /* successfully allocated. */
    if (DosAllocMem( &pBuf, cbNeeded,
                    PAG_READ|PAG_WRITE|PAG_COMMIT)

)
    {
        /* Set the count of bytes needed for the buffer to the value */
        /* returned in cbNeeded from the first call. */
        cbBuf = cbNeeded ;

        /* Make the call again with all the correct values. */
        SplQueryJob(pszComputerName,pszQueueName,ulJob,
                    ulLevel, pBuf, cbBuf, &cbNeeded) ;

        /* Set a pointer to point to the beginning of the buffer that holds */
        /* the returned structure. */
        pprj3=(PPRJINFO3)pBuf;

        /* Print out the information for each element in the structure. */
        printf("Job ID      = %d\n", pprj3->uJobId);
        printf("Job Priority= %d\n", pprj3->uPriority);
        printf("User Name   = %s\n", pprj3->pszUserName);
        printf("Position    = %d\n", pprj3->uPosition);
        printf("Status      = %d\n", pprj3->fsStatus);
        printf("Submitted   = %ld\n", pprj3->ulSubmitted);
        printf("Size        = %ld\n", pprj3->ulSize);
        printf("Comment     = %s\n", pprj3->pszComment);
        printf("Document    = %s\n", pprj3->pszDocument);
        printf("Notify Name = %s\n", pprj3->pszNotifyName);
        printf("Data Type   = %s\n", pprj3->pszDataType);
        printf("Parms       = %s\n", pprj3->pszParms);
        printf("Status      = %s\n", pprj3->pszStatus);
        printf("Queue       = %s\n", pprj3->pszQueue);
        printf("QProc Name  = %s\n", pprj3->pszQProcName);
        printf("QProc Parms = %s\n", pprj3->pszQProcParms);
        printf("Driver Name = %s\n", pprj3->pszDriverName);
        printf("Printer Name= %s\n", pprj3->pszPrinterName);

        /* If pDriverData is NULL, then we can not access any data. */
        if (pprj3->pDriverData)
        {
            printf(" pDriverData->cb      - %ld\n",
                    (ULONG)pprj3->pDriverData->cb);
            printf(" pDriverData->lVersion - %ld\n",
                    (ULONG)pprj3->pDriverData->lVersion) ;
            printf(" pDriverData->szDeviceName - %s\n",
                    pprj3->pDriverData->szDeviceName) ;
        }
        printf("/n");
    }
}

```

SplQueryJob – Spooler Query Job

```
        /* Free memory that we allocated.                */
        DosFreeMem(pBuf) ;
    }
else
{
    /* If we are here than we have an error code. Print it out. */
    printf("SplQueryJob Error=%ld,Bytes Needed=%ld\n",splerr, cbNeeded);
}
DosExit( EXIT_PROCESS , 0 ) ;
return(splerr);
}
```


SplQueryQueue – Spooler Query Queue

```
#define INCL_SPL /* Or use INCL_PM */
```

SPLERR SplQueryQueue (PSZ pszComputerName, PSZ pszQueueName, ULONG ulLevel, PVOID pBuf, ULONG cbBuf, PULONG pcbNeeded)

This function supplies information about a print queue, and, optionally, about the jobs in it.

Parameters

- pszComputerName (PSZ)** – input
Name of computer where queue is to be queried.
A NULL string specifies the local workstation.
- pszQueueName (PSZ)** – input
Queue name.
- ulLevel (ULONG)** – input
Level of detail required.
This must be 3, 4, 5 or 6.
- pBuf (PVOID)** – input
Buffer.
- cbBuf (ULONG)** – input
Size, in bytes, of Buffer.
- pcbNeeded (PULONG)** – output
Size in bytes of available information.

Returns

Return

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
ERROR_MORE_DATA (234)	Additional data is available.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_BufTooSmall (2123)	The API return buffer is too small.
NERR_QNotFound (2150)	The printer queue does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

SplQueryQueue – Spooler Query Queue

Remarks

The buffer contents on return are:

<i>ulLevel</i>	Buffer Contents
3	A PRQINFO3 structure, with associated variable information up to <i>cbBuf</i> .
4	A PRQINFO3 structure, with associated variable information, and an array of PRJINFO2 structures, one for each job in the queue, up to <i>cbBuf</i> .
5	A queue name of type PSZ.
6	A PRQINFO6 structure, with associated variable information up to <i>cbBuf</i> .

If *ulLevel* is 3 or 4, and *pBuf* cannot hold the entire PRQINFO3 structure, SplQueryQueue returns NERR_BufTooSmall (2123). If *ulLevel* is 6, and *pBuf* cannot hold the entire PRQINFO6 structure, SplQueryQueue returns NERR_BufTooSmall (2123).

If *ulLevel* is 4, and *pBuf* cannot hold all the available PRJINFO2 structures, SplQueryQueue returns ERROR_MORE_DATA (234).

To obtain the size of buffer required, call SplQueryQueue with the required value of *ulLevel* and a NULL buffer. The number of bytes required is returned in *pcbNeeded*.

Related Functions

- SplEnumQueue
- SplQueryJob
- SplSetJob
- SplSetQueue

Example Code

This sample code queries the local workstation for a queue name that is entered at the command prompt. The query is done at level 4 which returns returns in the buffer information in a PRQINFO3 structure and follows this with PRJINFO2 structures - one for each job in the queue.

```
#define INCL_SPL
#define INCL_SPLDOSPRINT

#include <os2.h>
#include <stdio.h>
#include <neterr.h>

INT main (argc, argv)
    INT argc;
    CHAR *argv[];
{
    ULONG splerr ;
    ULONG cbBuf;
    ULONG cbNeeded ;
    ULONG ulLevel ;
    ULONG i ;
    USHORT uJobCount ;
    PSZ pszComputerName ;
    PSZ pszQueueName ;
    PVOID pBuf;
    PPRJINFO2 prj2 ;
    PPRQINFO3 prq3 ;

    if (argc != 2)
    {
        printf("Syntax: setqryq QueueName \n");
        DosExit( EXIT_PROCESS , 0 ) ;
    }

    pszComputerName = (PSZ)NULL ;
```

SplQueryQueue — Spooler Query Queue

```

pszQueueName = argv[1];
ulLevel = 4L;
splerr = SplQueryQueue(pszComputerName, pszQueueName, ulLevel,
                       (PVOID)NULL, 0L, &cbNeeded );
if (splerr != NERR_BufTooSmall || splerr != ERROR_MORE_DATA )
{
    printf("SplQueryQueue Error=%ld, cbNeeded=%ld\n", splerr, cbNeeded);
    DosExit( EXIT_PROCESS, 0 );
}
if (!DosAllocMem( &pBuf, cbNeeded,
                  PAG_READ|PAG_WRITE|PAG_COMMIT)
{
    cbBuf = cbNeeded;
    splerr = SplQueryQueue(pszComputerName, pszQueueName, ulLevel,
                           pBuf, cbBuf, &cbNeeded);

    prq3=(PPRQINF03)pBuf;
    printf("Queue info: name- %s\n", prq3->pszName);
    printf(" priority - %d starttime - %d endtime - %d fsType - %X\n",
           prq3->uPriority, prq3->uStartTime, prq3->uUntilTime,
           prq3->fsType );
    printf(" pszSepFile - %s\n", prq3->pszSepFile);
    printf(" pszPrProc - %s\n", prq3->pszPrProc);
    printf(" pszParms - %s\n", prq3->pszParms);
    printf(" pszComment - %s\n", prq3->pszComment);
    printf(" pszPrinters - %s\n", prq3->pszPrinters);
    printf(" pszDriverName - %s\n", prq3->pszDriverName);
    if (prq3->pDriverData)
    {
        printf(" pDriverData->cb - %ld\n",
               (ULONG)prq3->pDriverData->cb);
        printf(" pDriverData->lVersion - %ld\n",
               (ULONG)prq3->pDriverData->lVersion);
        printf(" pDriverData->szDeviceName - %s\n",
               prq3->pDriverData->szDeviceName);
    }
    /* Store the job count for use later in the for loop. */
    uJobCount = prq3->cJobs;
    printf("Job count in this queue is %d\n", uJobCount);

    /* Increment the pointer to the PRQINF03 structure so that it points to */
    /* the first structure after itself. */
    prq3++;

    /* Cast the prq3 pointer to point to a PRJINF02 structure, and set a */
    /* pointer to point to that place. */
    prj2=(PPRJINF02)prq3;
    for (i=0; i<uJobCount; i++) {
        printf("Job ID = %d\n", prj2->uJobId);
        printf("Priority = %d\n", prj2->uPriority);
        printf("User Name = %s\n", prj2->pszUserName);
        printf("Position = %d\n", prj2->uPosition);
        printf("Status = %d\n", prj2->fsStatus);
        printf("Submitted = %ld\n", prj2->ulSubmitted);
        printf("Size = %ld\n", prj2->ulSize);
        printf("Comment = %s\n", prj2->pszComment);
        printf("Document = %s\n", prj2->pszDocument);
    }
}

```

SplQueryQueue – Spooler Query Queue

```
        /* Increment the pointer to point to the next structure.          */
        prj2++;
    } /* endfor */
    DosFreeMem(pBuf) ;
    }
    DosExit( EXIT_PROCESS , 0 ) ;
    return (splerr);
}
```

SpiReleaseJob – Spooler Release Job

#define INCL_SPL /* Or use INCL_PM */

SPLERR SpiReleaseJob (PSZ pszComputerName, PSZ pszQueueName, ULONG ulJob)

This function releases a held print job.

Parameters

- pszComputerName (PSZ)** – input
Name of computer where job is to be continued.
A NULL string specifies the local workstation.
- pszQueueName (PSZ)** – input
Queue Name.
- ulJob (ULONG)** – input
Job identification number.

Returns

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_JobNotFound (2151)	The print job does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_JobInvalidState (2164)	This operation cannot be performed on the print job in its current state.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

Any job can be released by a user with administrator privilege.

A job created locally can be released locally regardless of user privilege level, but it can be released remotely only by a user with administrator privilege.

A remote job can be released by a user without administrator privilege only if the user identification of the person initiating the request is the same as the user identification of the person who created the job.

Related Functions

- SpiEnumJob
- SpiHoldJob
- SpiQueryJob

Example Code

This sample code will release the job id that is entered at the prompt.

```
#define INCL_BASE
#define INCL_SPL
#include <os2.h>
#include <stdio.h>      /* for printf function */
#include <stdlib.h>     /* for atoi function  */
#include <neterr.h>     /* for error codes   */

INT main (argc, argv)
    INT argc;
    CHAR *argv[];
{
    SPLERR splerr ;
    ULONG ulJob ;
    PSZ    pszComputerName = NULL ;
    PSZ    pszQueueName = NULL ;

    /* Get job id from the input argument */
    ulJob = atoi(argv[1]);

    /* Call the function to do the release. If an error is returned, print it. */
    splerr=SplReleaseJob( pszComputerName, pszQueueName, ulJob);
    switch (splerr)
    {
        case NO_ERROR:
            printf("Job %d was released.\n",ulJob);
            break;
        case NERR_JobNotFound:
            printf("Job does not exist.\n");
            break;
        case NERR_JobInvalidState:
            printf("This operation can't be performed on the print Job.\n");
            break;
        default:
            printf("Errorcode = %ld\n",splerr);
    } /* endswitch */
    DosExit( EXIT_PROCESS , 0 ) ;
    argc;
    return (splerr);
}
```

SplReleaseQueue — Spooler Release Queue

```
#define INCL_SPL /* Or use INCL_PM */
```

SPLERR SplReleaseQueue (PSZ pszComputerName, PSZ pszQueueName)

This function releases a held print queue.

Parameters

pszComputerName (PSZ) – input

Name of computer where queue is to be continued.

A NULL string specifies the local workstation.

pszQueueName (PSZ) – input

Queue name.

Returns

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_QNotFound (2150)	The printer queue does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

This function releases a queue that has been held by a SplHoldQueue function, or disabled by an error on the queue. It does not affect an active print queue.

To release a queue on a remote server requires administrator privilege on the remote server.

Related Functions

- SplEnumQueue
- SplHoldQueue
- SplQueryQueue

SplReleaseQueue – Spooler Release Queue

Example Code

This sample code will release the local queue that is entered at the prompt.

```
#define INCL_SPL
#include <os2.h>
#include <stdio.h>      /* for printf function */
#include <neterr.h>      /* for error codes */

INT main (argc, argv)
    INT argc;
    CHAR *argv[];
{
    SPLERR splerr ;
    PSZ    pszComputerName = NULL ;
    PSZ    pszQueueName ;

    /* Get queue name from the input argument. */
    pszQueueName = argv[1];

    /* Call the function to do the release. If an error is returned, print it. */
    splerr=SplReleaseQueue(pszComputerName, pszQueueName);
    if (splerr != 0L)
    {
        switch (splerr)
        {
            case NERR_QNotFound:
                printf("Queue does not exist.\n");
                break;
            case NERR_SpoolerNotLoaded:
                printf("The Spooler is not running.\n");
                break;
            default:
                printf("Errorcode = %ld\n",splerr);
        } /* endswitch */
    }
    else
    {
        printf("Queue %s was released.\n",pszQueueName);
    } /* endif */
    DosExit( EXIT_PROCESS , 0 ) ;
    return (splerr);
}
```


SplSetDevice – Spooler Set Device

```
#define INCL_SPL /* Or use INCL_PM */
```

SPLERR SplSetDevice (PSZ pszComputerName, PSZ pszPrintDeviceName, ULONG ulLevel, PVOID pBuf, ULONG cbBuf, ULONG ulParmNum)

This function modifies the configuration of a print device.

Parameters

pszComputerName (PSZ) – input

Name of computer where print device is to be modified.

A NULL string specifies the local workstation.

pszPrintDeviceName (PSZ) – input

Name of Print Device.

ulLevel (ULONG) – input

Level of detail required.

This must be 3.

pBuf (PVOID) – input

Buffer.

cbBuf (ULONG) – input

Size, in bytes, of Buffer.

ulParmNum (ULONG) – input

Parameter number.

Specifies either that the entire PRDINFO3 structure is to be modified, or that one specific parameter only is to be modified. If *ulParmNum* is 0, *pBuf* must contain a complete PRDINFO3 structure. Otherwise, *pBuf* must contain a valid value corresponding to the parameter to be modified:

Parameter	Constant (Value)
<i>pszLogAddr</i>	PRD_LOGADDR_PARMNUM (3)
<i>pszComment</i>	PRD_COMMENT_PARMNUM (7)
<i>pszDrivers</i>	PRD_DRIVERS_PARMNUM (8)
<i>usTimeout</i>	PRD_TIMEOUT_PARMNUM (10)

Returns

NO_ERROR (0)

No errors occurred.

ERROR_ACCESS_DENIED (5)

Access is denied.

ERROR_NOT_SUPPORTED (50)

This request is not supported by the network.

ERROR_BAD_NETPATH (53)

The network path cannot be located.

ERROR_INVALID_PARAMETER (87)

An invalid parameter is specified.

ERROR_INVALID_LEVEL (124)

The level parameter is invalid.

NERR_NetNotStarted (2102)

The network program is not started.

NERR_BufTooSmall (2123)

The API return buffer is too small.

NERR_DestNotFound (2152)

The print device cannot be found.

NERR_SpoolerNotLoaded (2161)

The spooler is not running.

NERR_DestInvalidState (2162)

This operation cannot be performed on the print device.

SplSetDevice — Spooler Set Device

NERR_SpoolNoMemory (2165)	A spooler memory allocation failure occurred.
NERR_DriverNotFound (2166)	The device driver does not exist.
NERR_BadDev (2341)	The requested device is invalid.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

This function allows modification of a print device and its connection to a logical address. To disconnect a print device from a port, use *ulLevel*=3, *ulParmNum*=3, and *pBuf* is a NULL string.

To modify a print device on a remote server requires administrator privilege.

Related Functions

- SplEnumDevice
- SplEnumDriver
- SplEnumPort
- SplEnumPrinter
- SplQueryDevice

Example Code

This sample code first gets a device name from the command line. It then prompts the user for a parameter number and a value associated with it.

```
#define INCL_BASE
#define INCL_DOSMEMMGR
#define INCL_SPL
#define INCL_SPLDOSPRINT
#include <os2.h>
#include <stdio.h>          /* for printf function */
#include <string.h>         /* for strlen function */
#include <stdlib.h>         /* for atoi function */
#include <neterr.h>         /* for error code */

INT main (argc, argv)
    INT argc;
    CHAR *argv[];
{
    CHAR    bufValue[2]={0};
    CHAR    bufInput[128]={0};
    ULONG   splerr ;
    ULONG   cbBuf ;
    ULONG   ulParmNum ;
    USHORT  usParm;
    PSZ     pszComputerName ;
    PSZ     pszPrintDeviceName ;
    PVOID   pBuf;

    if (argc != 2)
    {
        printf("Syntax: sdset DeviceName \n");
        DosExit( EXIT_PROCESS , 0 ) ;
    }
    pszComputerName = (PSZ)NULL ;

    /* Set the print device name to the value from the command line. */
    pszPrintDeviceName = argv[1];

    /* Get the parameter and the value. Store them in buffers. */
    printf("Enter Parameter number to be modified\n");
```

SplSetDevice — Spooler Set Device

```
gets(&bufValue[0]);
printf("Enter new parameter value \n");
gets(&bufInput[0]);

/* Convert the input parmnum to a ULONG. */
ulParmNum = atoi(&bufValue[0]);

switch (ulParmNum)
{
    case 10:
        /* Determine the size of the buffer. */
        cbBuf = sizeof(PUSHORT);

        /* Convert the input parameter to a USHORT. */
        usParm = (USHORT)atoi(&bufInput[0]);

        /* Point the buffer to the value. */
        pBuf = &usParm;
        break;
    case 3:
    case 7:
    case 8:
        /* Determine the size of the buffer. */
        cbBuf = strlen(&bufInput[0])+1;

        /* Point the buffer to the value. */
        pBuf = (PSZ)&bufInput;
        break;
    default:
        printf("Invalid number\n");
        DosExit( EXIT_PROCESS , 0 ) ;
        break;
}

/* Make the call. */
splerr = SplSetDevice(pszComputerName,pszPrintDeviceName,3L,
                    pBuf,cbBuf,ulParmNum) ;

/* Print out the result. */
printf("SplSetDevice Err= %ld, Parameter= %d, cbBuf= %ld ,ulParmNum= %ld\n",
        splerr, usParm, cbBuf, ulParmNum);

DosExit( EXIT_PROCESS , 0 ) ;
return (splerr);
}
```

```
#define INCL_SPL /* Or use INCL_PM */
```

```
SPLERR SplSetJob (PSZ pszComputerName, PSZ pszQueueName, ULONG ulJob,  
                  ULONG ulLevel, PVOID pBuf, ULONG cbBuf, ULONG ulParmNum)
```

This function modifies the instructions for a print job.

Parameters

pszComputerName (PSZ) – input

Name of computer where job is to be modified.

A NULL string specifies the local workstation.

pszQueueName (PSZ) – input

Queue Name.

ulJob (ULONG) – input

Job identification number.

ulLevel (ULONG) – input

Level of detail required.

This must be 3.

pBuf (PVOID) – input

Buffer.

cbBuf (ULONG) – input

Size, in bytes, of Buffer.

ulParmNum (ULONG) – input

Parameter number.

Specifies either that the entire PRJINFO3 structure is to be modified, or that one specified parameter only is to be modified.

If *ulParmNum* is 0, *pBuf* must contain a complete PRJINFO3 structure. Otherwise, *pBuf* must contain a valid value corresponding to the parameter to be modified, as follows:

Parameter	Value
<i>pszNotifyName</i>	PRJ_NOTIFYNAM_PARMNUM (3)
<i>pszDataType</i>	PRJ_DATATYPE_PARMNUM (4)
<i>pszParms</i>	PRJ_PARMS_PARMNUM (5)
<i>uPosition</i>	PRJ_POSITION_PARMNUM (6)
<i>pszComment</i>	PRJ_COMMENT_PARMNUM (11)
<i>pszDocument</i>	PRJ_DOCUMENT_PARMNUM (12)
<i>pszStatus</i>	PRJ_STATUSCOMMENT_PARMNUM (13)
<i>uPriority</i>	PRJ_PRIORITY_PARMNUM (14)
<i>pszQProcParms</i>	PRJ_PROCPARMS_PARMNUM (16)
<i>pDriverData</i>	PRJ_DRIVERDATA_PARMNUM (18)

uPosition must be given the appropriate value, as follows:

Value	Change
0	No change
1	Move to first place
>1	Move to this position, or if the specified value is greater than the number of jobs in the queue, move to the end of the queue.

SplSetJob — Spooler Set Job

Returns

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_BufTooSmall (2123)	The API return buffer is too small.
NERR_JobNotFound (2151)	The print job does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_JobInvalidState (2164)	This operation cannot be performed on the print job in its current state.
NERR_SpoolNoMemory (2165)	A spooler memory allocation failure occurred.
NERR_DriverNotFound (2166)	The device driver does not exist.
NERR_ProcNotFound (2168)	The queue processor is not installed.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

The job priority is changed, if necessary, to the priority of the next job after the new position. If the spooler is restarted, the order in which jobs are put on the queue depends on the priority and age of the job. This order may be different from the order following the SplSetJob call.

Users without administrator privilege can use SplSetJob only for jobs created when the same user name was logged on. They can move jobs backwards only, and cannot increase the job priority above the queue priority.

A job created locally has no associated user name, and any user can set information locally for the job. Only an administrator can set information for a job on a remote server.

Related Functions

- SplEnumJob
- SplQueryJob
- SplQueryQueue

Example Code

This sample code first gets a queue name and a jobid from the command prompt. It then prompts the user to enter a parameter number and a value for that number.

```
#define INCL_BASE
#define INCL_DOSMEMMGR
#define INCL_SPL
#define INCL_SPLDOSPRINT
#include <os2.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <neterr.h>

INT main (argc, argv)
    INT argc;
    CHAR *argv[];
{
    CHAR    bufValue[2]={0};
    CHAR    bufInput[128]={0};
    ULONG   splerr ;
    ULONG   cbBuf;
    ULONG   ulParmNum ;
    ULONG   ulJob ;
    USHORT  usParm;
    PSZ     pszComputerName ;
    PSZ     pszQueueName ;
    PVOID   pBuf;

    if (argc != 3)
    {
        printf("Syntax: sjset QueueName JobID \n");
        DosExit( EXIT_PROCESS , 0 ) ;
    }
    pszComputerName = (PSZ)NULL ;

    /* Set values to those entered at the prompt. */
    pszQueueName = argv[1] ;
    ulJob = atoi (argv[2]);

    /* Request a parameter and the associated value. Store them in buffers. */
    printf("Enter Parameter number to be modified\n");
    gets(&bufValue[0]);
    printf("Enter new parameter value \n");
    gets(&bufInput[0]);

    /* Convert the ParmNum to a ULONG. */
    ulParmNum = atoi(&bufValue[0]);
    switch (ulParmNum)
    {
        case 6:
        case 14:
            /* Calculate size of buffer needed if this is the parameter.*/
            cbBuf = sizeof(PUSHORT);

            /* Convert input parameter into a USHORT. */
            usParm =(USHORT)atoi(&bufInput[0]);

            /* Point pBuf to the value. */
            pBuf = &usParm;
            break;
        case 3:
        case 4:
        case 5:
```

SplSetJob — Spooler Set Job

```
case 11:
case 12:
case 16:
    /* Calculate size of buffer needed if this is the parameter.*/
    cbBuf = strlen(&bufInput[0])+1;

    /* Point pBuf to the value. */
    pBuf = (PSZ)&bufInput;
    break;
case 18:
    printf("In order to keep code simple, this is not implemented.");
    break;
default:
    printf("Invalid number\n");
}
splerr = SplSetJob(pszComputerName,pszQueueName,ulJob,3L,
                  pBuf,cbBuf,ulParmNum) ;
if ( !splerr)
    printf("Parameter was set.");
else
    printf("SplSetJob Error= %ld, Parameter= %d, Buf= %ld ,ParmNum= %ld\n",
          splerr, usParm, cbBuf, ulParmNum);
DosExit( EXIT_PROCESS , 0 ) ;
return (splerr) ;
}
```

SplSetQueue – Spooler Set Queue

```
#define INCL_SPL /* Or use INCL_PM */
```

**SPLERR SplSetQueue (PSZ pszComputerName, PSZ pszQueueName, ULONG ulLevel,
PVOID pBuf, ULONG cbBuf, ULONG IParmNum)**

This function modifies the configuration of a print queue.

Parameters

pszComputerName (PSZ) – input

Name of computer where queue is to be modified.

A NULL string specifies the local workstation.

pszQueueName (PSZ) – input

Queue name.

ulLevel (ULONG) – input

Level of detail required.

This must be 3 or 6.

pBuf (PVOID) – input

Buffer.

cbBuf (ULONG) – input

Size, in bytes, of Buffer.

IParmNum (ULONG) – input

Parameter number.

Specifies either that the entire PRQINFO3 or PRQINFO6 structure is to be modified, or that one specified parameter only is to be modified.

If *IParmNum* is 0, *pBuf* must contain a complete PRQINFO3 or PRQINFO6 structure. Otherwise, *pBuf* must contain a valid value corresponding to the parameter to be modified, as follows:

Parameter	Value
<i>uPriority</i>	PRQ_PRIORITY_PARMNUM (2)
<i>uStartTime</i>	PRQ_STARTTIME_PARMNUM (3)
<i>uUntilTime</i>	PRQ_UNTILTIME_PARMNUM (4)
<i>pszSepFile</i>	PRQ_SEPARATOR_PARMNUM (5)
<i>pszPrProc</i>	PRQ_PROCESSOR_PARMNUM (6)
<i>pszParms</i>	PRQ_PARMNS_PARMNUM (8)
<i>pszComment</i>	PRQ_COMMENT_PARMNUM (9)
<i>fsType</i>	PRQ_TYPE_PARMNUM (10)
<i>pszPrinters</i>	PRQ_PRINTERS_PARMNUM (12)
<i>pszDriverName</i>	PRQ_DRIVERNAME_PARMNUM (13)
<i>pDriverData</i>	PRQ_DRIVERDATA_PARMNUM (14)
<i>pszRemoteComputerName</i>	PRQ_REMOTE_COMPUTER_PARMNUM (15)
<i>pszRemoteQueueName</i>	PRQ_REMOTE_QUEUE_PARMNUM (16)

SplSetQueue — Spooler Set Queue

Returns

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
NERR_NetNotStarted (2102)	The network program is not installed.
NERR_RedirectedPath (2117)	The operation is invalid on a redirected resource.
NERR_BufTooSmall (2123)	The API return buffer is too small.
NERR_QNotFound (2150)	The printer queue does not exist.
NERR_DestNotFound (2152)	The printer destination cannot be found.
NERR_DestNoRoom (2157)	The maximum number of printer destinations has been reached.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_DestInvalidState (2162)	This operation cannot be performed on the print destination.
NERR_SpoolNoMemory (2165)	A spooler memory allocation failure occurred.
NERR_DriverNotFound (2166)	The device driver does not exist.
NERR_DataTypeInvalid (2167)	The datatype is not supported by the processor.
NERR_ProcNotFound (2168)	The queue processor is not installed.
NERR_BadDev (2341)	The requested device is invalid.
NERR_CommDevInUse (2343)	The requested device is invalid.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

If the *uPriority* field in PRQINFO3 or PRQINFO6 is set to PRQ_NO_PRIORITY, the queue priority is not changed.

Related Functions

- SplCreateQueue
- SplEnumDevice
- SplEnumDriver
- SplEnumQueue
- SplEnumQueueProcessor
- SplQueryQueue

SplSetQueue — Spooler Set Queue

Example Code

This sample code prompts the user to enter a parameter number and a value at the prompt. This value is then put into a buffer for use by the function.

```
#define INCL_SPL
#define INCL_SPLDOSPRINT

#include <os2.h>
#include <stdio.h> /* for printf function */
#include <stdlib.h> /* for atoi function */
#include <string.h> /* for strlen function */
#include <neterr.h> /* for error codes */

INT main (argc, argv)
    INT argc;
    CHAR *argv[];
{
    CHAR    bufValue[2] = {0};
    CHAR    bufInput[128] = {0};
    ULONG   splerr ;
    ULONG   cbBuf ;
    ULONG   ulParmNum ;
    USHORT  usParm ;
    PSZ     pszComputerName ;
    PSZ     pszQueueName ;
    PVOID   pBuf;

    if (argc != 2)
    {
        printf("Syntax: setqryq QueueName \n");
        DosExit( EXIT_PROCESS , 0 ) ;
    }

    /* This function will be for the local workstation.
    pszComputerName = (PSZ)NULL ;

    /* Get the parameter from the command line.
    pszQueueName = argv[1];

    /* Prompt the user for the parameter and values, and put them in buffers. */
    printf("Enter Parameter number to be modified\n");
    gets(&bufValue[0]);
    printf("Enter new parameter value \n");
    gets(&bufInput[0]);

    /* Convert the ParmNum to a ULONG.
    ulParmNum = atoi(&bufValue[0]);
    switch (ulParmNum){
        case 2:
        case 3:
        case 4:
        case 10:
            /* Determine the size of the buffer needed.
            cbBuf = sizeof(PUSHORT);

            /* Convert the buffer input to a USHORT.
            usParm =(USHORT)atoi(&bufInput[0]);

            /* Set the pBuf pointer to point to the value obtained.
            pBuf = &usParm;
            break;
        case 5:
        case 6:
        case 8:
```

SplSetQueue – Spooler Set Queue

```
case 9:
case 12:
case 13:
    /* Determine the size of the buffer needed. */
    cbBuf = strlen(&bufInput[0])+1;

    /* Set the pBuf pointer to point to the value obtained from input. */
    pBuf = (PSZ)&bufInput;
    break;
case 14:
    printf("For simplicity this is not implemented.");
    break;
default:
    printf("Invalid number\n");
    DosExit( EXIT_PROCESS , 0 ) ;
    break;
}
/* Make the call with all the proper parameters. */
splerr = SplSetQueue(pszComputerName, pszQueueName, 3L,
                    pBuf, cbBuf, ulParmNum) ;

/* Print the resultant error code, and the parameters entered. */
printf("SplSetQueue Error= %ld, Parameter= %d, cbBuf= %ld,
        ulParmNum= %ld\n",
        splerr, usParm, cbBuf, ulParmNum);
DosExit( EXIT_PROCESS , 0 ) ;
return (splerr);
}
```

Glossary

A

accelerator. A single key stroke that invokes an application-defined function.

accelerator table. Used to define which key strokes are treated as *accelerators* and the commands they are translated into.

access permission. All access rights that a user has regarding an object.

action. One of a set of defined tasks that a computer performs. Users request the application to perform an action in several ways, such as typing a command, pressing a function key, or selecting the action name from an action bar or menu.

action bar. The area at the top of a window that contains the choices currently available in the application program.

action point. The current position on the screen at which the pointer is pointing. (Contrast with *hot spot* and *input focus*.)

active program. A program currently running on the computer. See also *interactive program*, *noninteractive program*, and *foreground program*.

active window. The window with which the user is currently interacting.

address space. (1) The range of addresses available to a program. (2) The area of virtual storage available for a particular job.

alphanumeric video output. Output to the logical video buffer when the video adapter is in text mode and the logical video buffer is addressed by an application as a rectangular array of character cells.

anchor block. An area of Presentation Manager-internal resources allocated to a process or thread that calls WinInitialize.

anchor point. A point in a window used by a program designer or by a window manager to position a subsequently appearing window.

ANSI. American National Standards Institute.

APA. All points addressable.

API. Application programming interface. The formally-defined programming language that is between an IBM application program and the user of the program. See also *GPI*.

area. In computer graphics, a filled shape such as a solid rectangle.

ASCII. American National Standard Code for Information Interchange. A coded character set

consisting of 7-bit coded characters (8 bits including parity check), used for information interchange among data processing systems, data communications systems, and associated equipment.

ASCIIZ. A string of ASCII characters that is terminated with a byte containing the value 0.

aspect ratio. In computer graphics, the width-to-height ratio of an area, symbol, or shape.

asynchronous. (1) Without regular time relationship. (2) Unexpected or unpredictable with respect to the execution of a program's instructions. See also *synchronous*.

atom. A constant that represents a string. Once a string has been defined as an atom, the atom can be used in place of the string to save space. Strings are associated with their respective atoms in an *atom table*. See also *integer atom*.

atom table. Used to relate *atoms* with the strings that they represent. Also in the table is the mechanism by which the presence of a string can be checked.

attributes. Characteristics or properties that can be controlled, usually to obtain a required appearance; for example, the color of a line. See also *graphics attributes* and *segment attributes*.

AVIO. Advanced Video Input/Output.

B

background color. The color in which the background of a graphic primitive is drawn.

background mix. An attribute that determines how the background of a graphic primitive is combined with the existing color of the graphics presentation space. Contrast with *mix*.

background program. In multiprogramming, a program that executes with a low priority. Contrast with *foreground program*.

Bézier curves. A mathematical technique of specifying smooth continuous lines and surfaces, which require a starting point and a finishing point with several intermediate points that influence or control the path of the linking curve. Named after Dr. P. Bézier.

bit map. A representation in memory of the data displayed on an APA device, usually the screen.

block. (1) A string of data elements recorded or transmitted as a unit. The elements may be characters, words, or logical records. (2) To combine two or more data elements in one block.

border. A visual indication (for example, a separator line or a background color) of the boundaries of a window.

breakpoint. (1) An instruction in a program for halting execution. Breakpoints are usually established at positions in a program where halts, caused by external intervention, are convenient for restarting. (2) A place in a program, specified by a command or a condition, where the system halts execution and gives control to the workstation user or to a specified program.

bucket. One or more fields in which the result of an operation is kept.

buffer. (1) A portion of storage used to hold input or output data temporarily. (2) To allocate and schedule the use of buffers.

button. A mechanism on a *pointing device*, such as a mouse, used to request or initiate an action. Contrast with *pushbutton* and *radio button*.

C

cache. A high-speed buffer storage that contains frequently accessed instructions and data; it is used to reduce access time.

cached micro presentation space. A presentation space from a Presentation Manager-owned store of micro presentation spaces. It can be used for drawing to a window only, and must be returned to the store when the task is complete.

call. (1) The action of bringing a computer program, a routine, or a subroutine into effect, usually by specifying the entry conditions and jumping to an entry point. (2) To transfer control to a procedure, program, routine, or subroutine.

calling order. A sequence of instructions together with any associated data necessary to perform a call. Also known as *calling sequence*.

cancel. An action that removes the current window or menu without processing it, and returns the previous window.

CASE statement. In C, provides the body of a window procedure. There is one CASE statement for each message type written to take specific actions.

cell. See *character cell*.

CGA. Color graphics adapter.

chained list. A list in which the data elements may be dispersed but in which each data element contains information for locating the next. Synonym for *linked list*.

character. A letter, digit, or other symbol.

character box. In computer graphics, the boundary that defines, in world coordinates, the horizontal and vertical space occupied by a single character from a character set. See also *character mode*. Contrast with *character cell*.

character cell. The physical, rectangular space in which any single character is displayed on a screen or printer device. Position is addressed by row and column coordinates. Contrast with *character box*.

character code. The means of addressing a character in a character set, sometimes called *code point*.

character mode. The character mode, in conjunction with the font type, determines the extent to which graphics characters are affected by the character box, shear, and angle attributes.

check box. A control window, shaped like a square button on the screen, that can be in a checked or unchecked state. It is used to select one or more items from a list. Contrast with *radio button*.

check mark. The symbol that is used to indicate a selected item on a pull-down.

child process. A process that is loaded and started by another process. Contrast with *parent process*.

child window. A window that is positioned relative to another window (either a main window or another child window). Contrast with *parent window*.

choice. An option that can be selected. The choice can be presented as text, as a symbol (number or letter), or as an icon (a pictorial symbol).

class. See *window class*.

class style. The set of properties that apply to every window in a window class.

client area. The area in the center of a window that contains the main information of the window.

clipboard. An area of main storage that can hold data being passed from one PM application to another. Various data formats can be stored.

clipping. In computer graphics, removing those parts of a display image that lie outside a given boundary.

clip limits. The area of the paper that can be reached by a printer or plotter.

clipping path. A clipping boundary in world-coordinate space.

CLOCK\$. Character-device name reserved for the system clock.

code page. An assignment of graphic characters and control-function meanings to all code points.

code point. Synonym for *character code*.

code segment. An executable section of programming code within a load module.

color dithering. See *dithering*.

command. The name and parameters associated with an action that a program can perform.

command area. An area composed of a command field prompt and a command entry field.

command entry field. An entry field in which users type commands.

command line. On a display screen, a display line usually at the bottom of the screen, in which only commands can be entered.

command prompt. A field prompt showing the location of the command entry field in a panel.

Common Programming Interface (CPI). A consistent set of specifications for languages, commands, and calls to enable applications to be developed across all SAA environments. See also *Systems Application Architecture*.

Common User Access (CUA). A set of rules that define the way information is presented on the screen, and the techniques for the user to interact with the information.

compile. To translate a program written in a higher-level programming language into a machine language program.

COM1, COM2, COM3. Character-device names reserved for serial ports 1 through 3.

CON. Character-device name reserved for the console keyboard and screen.

contiguous. Touching or joining at a common edge or boundary, for example, an unbroken consecutive series of storage locations.

control. The means by which an operator gives input to an application. A *choice* corresponds to a control.

Control Panel. In PM, a program used to set up user preferences that act globally across the system.

Control Program. The basic function of OS/2, including DOS emulation and the support for keyboard, mouse, and video input/output.

control window. A class of window used to handle a specific kind of user interaction. Radio buttons and check boxes are examples.

correlation. The action of determining which element or object within a picture is at a given position on the display. This follows a *pick* operation.

CPI. Common Programming Interface.

critical extended attribute. An extended attribute that is necessary for the correct operation of the system or a particular application.

CUA. Common User Access.

current position. The point from which the next primitive will be drawn.

cursor. A symbol displayed on the screen and associated with an input device. The cursor indicates where input from the device will be placed. Types of cursors include text cursors, graphics cursors, and selection cursors. Contrast with *pointer* and *input focus*.

D

data structure. (ISO) The syntactic structure of symbolic expressions and their storage-allocation characteristics.

DBCS. See *double-byte character set*.

deadlock. (1) Unresolved contention for the use of a resource. (2) An error condition in which processing cannot continue because each of two elements of the process is waiting for an action by, or a response from, the other. (3) An impasse that occurs when multiple processes are waiting for the availability of a resource that will not become available because it is being held by another process that is in a similar wait state.

debug. To detect, diagnose, and eliminate errors in programs.

decipoint. In printing, one tenth of a point. There are 72 points in an inch.

default procedure. Function provided by the Presentation Interface that may be used to process standard messages from dialogs or windows.

default value. A value used when no value is explicitly specified by the user. For example, in the graphics programming interface, the default line-type is 'solid'.

descendant. A process or session that is loaded and started by a parent process or parent session.

Desktop Manager. In PM, a window that displays a list of groups of programs, each of which can be started or stopped.

desktop window. The window, corresponding to the physical device, against which all other types of windows are established.

device context. A logical description of a data destination such as memory, metafile, display, printer, or plotter. See also *direct device context*, *information device context*, *memory device context*, *metafile device context*, *queued device context*, and *screen device context*.

device driver. A file that contains the code needed to attach and use a device such as a display, printer, or plotter.

device space. Coordinate space in which graphics are assembled after all GPI transformations have been applied. Device space is defined in device-specific units.

dialog. The interchange of information between a computer and its user through a sequence of requests by the user and the presentation of responses by the computer.

dialog box. A type of window that contains one or more controls for the formatted display and entry of data. Also known as a *pop-up window*. A modal dialog box is used to implement a pop-up window.

Dialog Box Editor. A WYSIWYG editor that creates dialog boxes for communicating with the application user.

dialog item. A component (for example, a menu or a button) of a dialog box. Dialog items are also used when creating dialog templates.

dialog tag language. A markup language used by the DTL compiler to create dialog objects.

dialog template. The definition of a dialog box, which contains details of its position, appearance, and window ID, and the window ID of each of its child windows.

direct device context. A logical description of a data destination that is a device other than the screen (for example, a printer or plotter), and where the output is not to go through the spooler. Its purpose is to satisfy queries. See also *device context*.

direct manipulation. The action of using the mouse to move objects around the screen. For example, moving files and directories around in the *File Manager*.

direct memory access (DMA). The transfer of data between main storage and input/output devices without intervention by the processor.

directory. A type of file containing the names and controlling information for other files or other directories.

display point. Synonym for *pel*.

dithering. The process used in color displays whereby every other pel is set to one color, and the intermediate pels are set to another. Together they produce the effect of a third color at normal viewing distances. This process can only be used on solid areas of color; it does not work on narrow lines, for example.

DMA. Direct memory access.

double-byte character set (DBCS). A set of characters in which each character is represented by two bytes. Languages such as Japanese, Chinese, and Korean, which contain more characters than can be represented by 256 code points, require double-byte character sets. Since each character requires two bytes, the entering, displaying, and printing of DBCS characters requires hardware and software that can support DBCS.

doubleword. A contiguous sequence of bits or characters that comprises two computer words and is capable of being addressed as a unit.

dragging. In computer graphics, moving an object on the display screen as if it were attached to the pointer.

drawing chain. See *segment chain*.

drop. To fix the position of an object that is being dragged, by releasing the select button of the pointing device.

DTL. See *dialog tag language*.

dual-boot function. A feature of OS/2 that allows the user to start DOS from within OS/2, or OS/2 from within DOS.

duplex. Pertaining to communication in which data can be sent and received at the same time. Synonymous with *full duplex*.

dynamic linking. The process of resolving external references in a program module at load time or run time rather than during linking.

dynamic-link library. A collection of executable programming code and data that is bound to an application at load time or run time, rather than during linking. The programming code and data in a dynamic link library can be shared by several applications simultaneously.

dynamic-link module. A module that is linked at load time or run time.

dynamic segments. Graphics segments drawn in exclusive-OR mix mode so that they can be moved from one screen position to another without affecting the rest of the displayed picture.

dynamic storage. (1) A device that stores data in a manner that permits the data to move or vary with time such that the specified data is not always available for recovery. (2) A storage in which the cells require repetitive application of control signals in order to retain stored data. Such repetitive application of the control signals is called a refresh operation. A dynamic storage may use static addressing or sensing circuits. (3) See also *static storage*.

E

EBCDIC. Extended binary-coded decimal interchange code. A coded character set consisting of 8-bit coded characters (9 bits including parity check), used for information interchange among data processing systems, data communications systems, and associated equipment.

EGA. Extended graphics adapter.

8.3 file-name format. A file-naming convention in which file names are limited to eight characters before and three characters after a single dot. Usually pronounced "eight-dot-three." See also *non-8.3 file-name format*.

element. An entry in a graphics segment that comprises one or more graphics orders and that is addressed by the element pointer.

entry field. An area on the screen, usually highlighted in some manner, in which users type information.

entry-field control. The means by which the application receives data entered by the user in an entry field. When it has the input focus, it displays a flashing pointer at the position where the next typed character will go.

entry panel. A defined panel type containing one or more entry fields and protected information such as headings, prompts, and explanatory text.

exception. An abnormal condition such as an I/O error encountered in processing a data set or a file.

exclusive system semaphore. A system semaphore that can be modified only by threads within the same process.

exit. The action that terminates the current function and returns the user to a higher level function. Repeated exit requests return the user to the point from which all functions provided to the system are accessible. Contrast with *cancel*.

extended attribute. An additional piece of information about a file object, such as its data format or category. It consists of a name and a value. A file object may have more than one extended attribute associated with it.

extended-choice selection. A mode that allows the user to select more than one item from a window. Not all windows allow extended choice selection. Contrast with *multiple-choice selection*.

extended help. A facility that provides users with information about an entire application panel rather than a particular item on the panel.

extent. Continuous space on a disk or diskette that is occupied by or reserved for a particular data set, data space, or file.

F

family-mode application. An application program that can run in the OS/2 environment and in the DOS environment. However, it cannot take advantage of many of the OS/2-mode facilities, such as multitasking, interprocess communication, and dynamic linking.

FAT. File allocation table.

FEA. Full extended attribute.

field-level help. Information specific to the field on which the cursor is positioned. This help function is "contextual" because it provides information about a specific item as it is currently used; the information is dependent upon the context within the work session.

file. A named set of records stored or processed as a unit.

file allocation table (FAT). In IBM personal computers, a table used by the operating system to allocate space on a disk for a file, and to locate and chain together parts of the file that may be scattered on different sectors so that the file can be used in a random or sequential manner.

file attribute. Any of the attributes that describe the characteristics of a file.

File Manager. In PM, a program that displays directories and files, and allows various actions on them.

file specification. The full identifier for a file, which includes its drive designation, path, file name, and extension.

file system driver (FSD). A program that manages file I/O and controls the format of information on the storage media.

fillet. A curve that is tangential to the end points of two adjoining lines. See also *polyfillet*.

flag. (1) An indicator or parameter that shows the setting of a switch. (2) A character that signals the occurrence of some condition, such as the end of a word.

focus. See *input focus*.

font. A particular size and style of typeface that contains definitions of character sets, marker sets, and pattern sets.

foreground program. The program with which the user is currently interacting. Also known as *interactive program*. Contrast with *background program*.

frame. The part of a window that can contain several different visual elements specified by the application, but drawn and controlled by PM. The frame encloses the client area.

frame styles. Different standard window layouts provided by PM.

FSD. File system driver.

full duplex. Synonym for *duplex*.

full-screen application. An application program that occupies the whole screen.

function. (1) In a programming language, a block, with or without formal parameters, whose execution is invoked by means of a call. (2) A set of related control statements that cause one or more programs to be performed.

function key. A key that causes a specified sequence of operations to be performed when it is pressed, for example, F1 and Alt-K.

function key area. The area at the bottom of a window that contains function key assignments such as F1=Help.

G

GDT. Global Descriptor Table.

general protection fault. An exception condition that occurs when a process attempts to use storage or a module that has some level of protection assigned to it, such as I/O privilege level. See also *IOPL code segment*.

Global Descriptor Table (GDT). Defines code and data segments available to all tasks in an application.

global dynamic-link module. A dynamic-link module that can be shared by all processes in the system that refer to the module name.

global file-name character. A special character used to refer to a set of file objects with a common base name. The asterisk (*) and question mark (?) are used as global file-name characters. For example, *.EXE can be used to refer to a set of files with the extension EXE.

glyph. A graphic symbol whose appearance conveys information.

GPI. Graphics programming interface. The formally-defined programming language that is between an IBM graphics program and the user of the program. See also *API*.

graphics. A picture defined in terms of graphic primitives and graphics attributes.

graphics attributes. Attributes that apply to graphic primitives. Examples are color, line type, and shading-pattern definition. See also *segment attributes*.

graphics field. The clipping boundary that defines the visible part of the presentation-page contents.

graphics model space. The conceptual coordinate space in which a picture is constructed after any model transforms have been applied. Also known as *model space*.

graphic primitive. A single item of drawn graphics, such as a line, arc, or graphics text string. See also *graphics segment*.

graphics segment. A sequence of related graphic primitives and graphics attributes. See also *graphic primitive*.

graying. The indication that a choice on a pull-down is unavailable.

group. A collection of logically-connected controls. For example, the buttons controlling paper size for a printer. See also *program group*.

H

handle. An identifier that represents an object, such as a device or window, to the Presentation Interface.

hard error. An error condition on a network that requires either that the system be reconfigured, or that the source of the error be removed before the system can resume reliable operation.

header. (1) System-defined control information that precedes user data. (2) The portion of a message that contains control information for the message, such as one or more destination fields, name of the originating station, input sequence number, character string indicating the type of message, and priority level for the message.

help. A function that provides information about a specific field, an application panel, or information about the help facility.

help index. A facility that allows the user to select topics for which help is available.

help panel. A panel with information to assist users that is displayed in response to a help request from the user.

help window. A Common User Access-defined secondary window that displays information when the user requests help.

heap. An area of free storage available for dynamic allocation by an application. Its size varies according to the storage requirements of the application.

hit testing. The means of identifying which window is associated with which input device event.

hook. A mechanism by which procedures are called when certain events occur in the system. For example,

the filtering of mouse and keyboard input before it is received by an application program.

hook chain. A sequence of hook procedures that are "chained" together so that each event is passed, in turn, to each procedure in the chain.

hot spot. The part of the pointer that must touch an object before it can be selected. This is usually the tip of the pointer. Contrast with *action point*.

I

icon. A pictorial representation of an item the user can select. Icons can represent items (such as a document file) that the user wants to work on, and actions that the user wants to perform. In PM, icons are used for data objects, system actions, and minimized programs.

icon area. In PM, the area at the bottom of the screen that is normally used to display the icons for minimized programs.

Icon Editor. The Presentation Manager-provided tool for creating icons.

image font. A set of symbols, each of which is described in a rectangular array of pels. Some of the pels in the array are set to produce the image of the symbol. Contrast with *outline font*.

information device context. A logical description of a data destination other than the screen (for example, a printer or plotter), but where no output will occur. Its purpose is to satisfy queries. See also *device context*.

information panel. A defined panel type characterized by a body containing only protected information.

input focus. The area of the screen that will receive input from an input device (typically the keyboard).

input router. An internal OS/2 process that removes messages from the system queue.

integer atom. A special kind of *atom* that represents a predefined system constant and carries no storage overhead. For example, names of window classes provided by PM are expressed as integer atoms.

interactive graphics. Graphics that can be moved or manipulated by a user at a terminal.

interactive program. A program that is running (active) and is ready to receive (or is receiving) input from the user. Compare with *active program* and contrast with *noninteractive program*.

Also known as a *foreground program*.

interchange file. Data that can be sent from one Presentation Interface application to another.

interval timer. (1) A timer that provides program interruptions on a program-controlled basis. (2) An electronic counter that counts intervals of time under program control.

IOctl. A device-specific command that requests a function of a device driver through the *DosDevIOctl* function.

I/O operation. An input operation to, or output operation from a device attached to a computer.

IOPL. Input/output privilege level.

IOPL code segment. An IOPL executable section of programming code that enables an application to directly manipulate hardware interrupts and ports without replacing the device driver. See also *privilege level*.

J

journal. A special-purpose file that is used to record changes made in the system.

K

Kanji. A graphic character set used in Japanese ideographic alphabets.

KBD\$. Character-device name reserved for the keyboard.

kernel. The part of an operating system that performs basic functions, such as allocating hardware resources.

kerning. The design of graphics characters so that their character boxes overlap. Used to space text proportionally.

keys help. A facility that gives users a listing of all the key assignments for the current application.

L

label. In a graphics segment, an identifier of one or more elements that is used when editing the segment.

language support procedure. Function provided by the Presentation Interface for applications that do not, or cannot (as in the case of COBOL and FORTRAN programs), provide their own dialog or window procedures.

LDT. Local Descriptor Table.

LIFO stack. A data stack from which data is retrieved in last-in, first-out order.

linked list. Synonym for *chained list*.

list box. A control window containing a vertical list of selectable descriptions.

list panel. A defined panel type that displays a list of items from which users can select one or more choices and then specify one or more actions to work on those choices.

load-on-call. A function of a linkage editor that allows selected segments of the module to be disk resident while other segments are executing. Disk resident segments are loaded for execution and given control when any entry point that they contain is called.

load time. The point in time at which a program module is loaded into main storage for execution.

local area network (LAN). A data network located on the user's premises in which serial transmission is used for direct data communication among data stations.

Local Descriptor Table (LDT). Defines code and data segments specific to a single task.

lock. A serialization mechanism by means of which a resource is restricted for use by the holder of the lock.

LPT1, LPT2, LPT3. Character-device names reserved for parallel printers 1 through 3.

M

main window. The window that is positioned relative to the *desktop window*.

map. (1) A set of values having a defined correspondence with the quantities or values of another set. (2) To establish a set of values having a defined correspondence with the quantities or values of another set.

marker box. In computer graphics, the boundary that defines, in world coordinates, the horizontal and vertical space occupied by a single marker from a marker set.

marker symbol. A symbol centered on a point. Graphs and charts can use marker symbols to indicate the plotted points.

maximize. A window-sizing action that makes the window the largest size possible.

media window. The part of the physical device (display, printer, or plotter) on which a picture is presented.

memory device context. A logical description of a data destination that is a memory bit map. See also *device context*.

memory management. A feature of the operating system for allocating, sharing, and freeing main storage.

menu. A type of panel that consists of one or more selection fields. Also called a *menu panel*.

message. (1) In PM, a packet of data used for communication between the Presentation Interface and windowed applications. (2) In a user interface, information not requested by users but presented to users by the computer in response to a user action or internal process.

message filter. The means of selecting which messages from a specific window will be handled by the application.

message queue. A sequenced collection of messages to be read by the application.

metafile. The generic name for the definition of the contents of a picture. Metafiles are used to allow pictures to be used by other applications.

metafile device context. A logical description of a data destination that is a metafile, which is used for graphics interchange. See also *device context*.

metalanguage. A language used to specify another language. For example, data types can be described using a metalanguage so as to make the descriptions independent of any one computer language.

mickey. A unit of measurement for physical mouse motion whose value depends on the mouse device driver currently loaded.

micro presentation space. A graphics presentation space in which a restricted set of the GPI function calls is available.

minimize. A window-sizing action that makes the window the smallest size possible. In PM, minimized windows are represented by icons.

mix. An attribute that determines how the foreground of a graphic primitive is combined with the existing color of graphics output. Also known as *foreground mix*. Contrast with *background mix*.

mixed character string. A string containing a mixture of one-byte and *Kanji* or Hangeul (two-byte) characters.

mnemonic. A method of selecting an item on a pull-down by means of typing the highlighted letter in the menu item.

modal dialog box. The type of control that allows the operator to perform input operations on only the current dialog box or one of its child windows. Also known as a *serial dialog box*. Contrast with *parallel dialog box*.

modeless dialog box. The type of control that allows the operator to perform input operations on any of the application's windows. Also known as a *parallel dialog box*. Contrast with *modal dialog box*.

model space. See *graphics model space*.

module definition file. A file that describes the code segments within a load module. For example, it indicates whether a code segment is loadable before module execution begins (preload), or loadable only when referred to at run time (load-on-call).

mouse. A hand-held device that is moved around to position the pointer on the screen.

MOUSE\$. Character-device name reserved for a mouse.

multiple-choice selection. A mode that allows users to select any number of choices, including none at all. See also *check box*. Contrast with *extended-choice selection*.

multitasking. The concurrent processing of applications or parts of applications. A running application and its data are protected from other concurrently running applications.

N

named pipe. A named buffer that provides client-to-server, server-to-client, or full duplex communication between unrelated processes. Contrast with *unnamed pipe*.

noncritical extended attribute. An extended attribute that is not necessary for the function of an application.

nondestructive read. A read process that does not erase the data in the source location.

non-8.3 file-name format. A file-naming convention in which path names can consist of up to 255 characters. See also *8.3 file-name format*.

noninteractive program. A program that is running (active) but is not ready to receive input from the user. Compare with *active program*, and contrast with *interactive program*.

nonretained graphics. Graphic primitives that are not remembered by the Presentation Interface once they have been drawn. Contrast with *retained graphics*.

NUL. Character-device name reserved for a nonexistent (dummy) device.

null-terminated string. A string of (n + 1) characters where the (n + 1)th character is the 'null' character (X'00'), and is used to represent an n-character string with implicit length. Also known as 'zero-terminated' string and 'ASCIIZ' string.

O

object window. A window that does not have a parent, but which may have child windows. An object window cannot be presented on a device.

open. To start working with a file, directory, or other object.

outline font. A set of symbols, each of which is created as a series of lines and curves. Synonymous with *vector font*. Contrast with *image font*.

output area. The area of the output device within which the picture is to be displayed, printed, or plotted.

owner window. A window into which specific events that occur in another (owned) window are reported.

owning process. The process that owns the resources that may be shared with other processes.

P

page. A 4KB segment of contiguous physical memory.

page viewport. A boundary in device coordinates that defines the area of the output device in which graphics are to be displayed. The presentation-page contents are transformed automatically to the page viewport in device space.

paint. The action of drawing or redrawing the contents of a window.

panel. A particular arrangement of information grouped together for presentation to the user in a window.

panel area. An area within a panel that contains related information. The three major Common User Access-defined panel areas are the action bar, the function key area, and the panel body.

panel body. The portion of a panel not occupied by the action bar, function key area, title or scroll bars. The panel body may contain protected information, selection fields, and entry fields. The layout and content of the panel body determine the panel type.

panel body area. The part of a window not occupied by the action bar or function key area. The panel body area may contain information, selection fields, and entry fields. Also known as *client area*.

panel body area separator. A line or color boundary that provides users with a visual distinction between two adjacent areas of a panel.

panel definition. A description of the contents and characteristics of a panel. A panel definition is the application developer's mechanism for predefining the format to be presented to users in a window.

panel ID. A panel element located in the upper left-hand corner of a panel body that identifies that particular panel within the application.

panel title. A panel element that identifies the information in the panel.

paper size. The size of paper, defined in either standard U.S. or European names (for example, A, B, A4), and measured in inches or millimeters respectively.

parallel dialog box. See *modeless dialog box*.

parent process. A process that loads and starts other processes. Contrast with *child process*.

parent window. The window relative to which one or more child windows are positioned. Contrast with *child window*.

partition. (1) A fixed-size division of storage. (2) On an IBM personal computer fixed disk, one of four possible storage areas of variable size; one may be accessed by DOS, and each of the others may be assigned to another operating system.

path. The part of a file specification that lists a series of directory names. Each directory name is separated by the backslash character. In the file specification C:\MYFILES\MISC\GLOSSARY.SCR, the path consists of MYFILES\MISC\.

pel. The smallest area of a display screen capable of being addressed and switched between visible and invisible states. Synonym for *display point*, *pixel*, and *picture element*.

pick. To select part of a displayed object using the pointer.

picture chain. See *segment chain*.

picture element. Synonym for *pel*.

PID. Process identification.

pipe. A named or unnamed buffer used to pass data between processes. A process reads from or writes to a pipe as if the pipe were a standard-input or

standard-output file. See also *named pipe* and *unnamed pipe*.

pixel. Synonym for *pel*.

plotter. An output device that uses pens to draw its output on paper or on transparency foils.

PM. Presentation Manager.

pointer. (1) The symbol displayed on the screen that is moved by a pointing device, such as a *mouse*. The pointer is used to point at items that users can select. Contrast with *cursor*. (2) A data element that indicates the location of another data element.

POINTERS\$. Character-device name reserved for a pointer device (mouse screen support).

pointing device. A device (such as a mouse) used to move a pointer on the screen.

pointings. Pairs of x-y coordinates produced by an operator defining positions on a screen with a pointing device, such as a *mouse*.

polyfillet. A curve based on a sequence of lines. It is tangential to the end points of the first and last lines, and tangential also to the midpoints of all other lines. See also *fillet*.

polyline. A sequence of adjoining lines.

pop. To retrieve an item from a last-in-first-out stack of items. Contrast with *push*.

pop-up window. A window that appears on top of another window in a dialog. Each pop-up window must be completed before returning to the underlying window.

Presentation Manager (PM). The visual component of OS/2 that presents, in windows, a graphics-based interface to applications and files installed and running in OS/2.

presentation page. The coordinate space in which a picture is assembled for display.

presentation space (PS). Contains the device-independent definition of a picture.

primary window. The window in which the main dialog between the user and the application takes place. In a multiprogramming environment, each application starts in its own primary window. The primary window remains for the duration of the application, although the panel displayed will change as the user's dialog moves forward. See also *secondary window*.

primitive. See *graphic primitive*.

primitive attribute. A specifiable characteristic of a graphic primitive. See *graphics attributes*.

print job. The result of sending a document or picture to be printed.

Print Manager. In PM, the part of the spooler that manages the spooling process. It also allows users to view print queues and to manipulate print jobs.

privilege level. A protection level imposed by the hardware architecture of the IBM personal computer. There are four privilege levels (number 0 through 3). Only certain types of programs are allowed to execute at each privilege level. See also *IOPL code segment*.

procedure call. In programming languages, a language construct for invoking execution of a procedure.

process. An instance of an executing application and the resources it is using.

program details. Information about a program that is specified in the *Program Manager* window and is used when the program is started.

program group. In PM, several programs that can be acted upon as a single entity.

program name. The full file specification of a program. Contrast with *program title*.

program title. The name of a program as it is listed in the *Program Manager* window. Contrast with *program name*.

prompt. A displayed symbol or message that requests input from the user or gives operational information. The user must respond to the prompt in order to proceed.

protocol. A set of semantic and syntactic rules that determines the behavior of functional units in achieving communication.

pseudocode. An artificial language used to describe computer program algorithms without using the syntax of any particular programming language.

pull-down. An *action bar* extension that displays a list of choices available for a selected action bar choice. After users select an action bar choice, the pull-down appears with the list of choices. Additional *pop-up windows* may appear from pull-down choices to further extend the actions available to users.

push. To add an item to a last-in-first-out stack of items. Contrast with *pop*.

pushbutton. A control window, shaped like a round-cornered rectangle and containing text, that invokes an immediate action, such as 'enter' or 'cancel'.

Q

queue. A linked list of elements waiting to be processed. For example, a queue may be a list of print jobs waiting to be printed.

queued device context. A logical description of a data destination (for example, a printer or plotter) where the output is to go through the spooler. See also *device context*.

R

radio button. A control window, shaped like a round button on the screen, that can be in a checked or unchecked state. It is used to select a single item from list. Contrast with *check box*.

RAS. Reliability, availability, and serviceability.

raster. (1) In computer graphics, a predetermined pattern of lines that provides uniform coverage of a display space. (2) The coordinate grid that divides the display area of a display device.

read-only file. A file that may be read from but not written to.

realize. To cause the system to ensure, wherever possible, that the physical color table of a device is set to the closest possible match in the logical color table.

recursive routine. A routine that can call itself or be called by another routine called by the recursive routine.

reentrant. The attribute of a program or routine that allows the same copy of the program or routine to be used concurrently by two or more tasks.

reference phrase. A word or phrase that is emphasized in a device-dependent manner to inform the user that additional information for the word or phrase is available.

reference phrase help. Provides help information for a selectable word or phrase.

refresh. To update a window, with changed information, to its current status.

region. A clipping boundary in device space.

register. A storage device having a specified storage capacity such as a bit, byte, or computer word, and usually intended for a special purpose.

remote file system. A file-system driver that gains access to a remote system without a block device driver.

resource. The means of providing extra information used in the definition of a window. A resource can contain definitions of fonts, templates, accelerators, and mnemonics; the definitions are held in a resource file.

resource file. A file containing information used in the definition of a window. Definitions can be of fonts, templates, accelerators, and mnemonics.

restore. To return a window to its original size or position following a sizing or moving action.

retained graphics. Graphic primitives that are remembered by the Presentation Interface after they have been drawn. Contrast with *nonretained graphics*.

return code. (1) A code used to influence the execution of succeeding instructions. (2) A value returned to a program to indicate the results of an operation requested by that program.

reverse video. A form of alphanumeric highlighting for a character, field, or cursor, in which its color is

exchanged with that of its background. For example, changing a red character on a black background to a black character on a red background.

RGB. Red-green-blue. For example, "RGB display".

roman. Relating to a type style with upright characters.

root segment. In a hierarchical database, the highest segment in the tree structure.

run time. (1) Any instant at which a program is being executed. (2) The time during which an instruction in an instruction register is decoded and performed.

S

SAA. Systems Application Architecture.

scheduler. A computer program designed to perform functions such as scheduling, initiation, and termination of jobs.

screen. The physical surface of a work station or terminal upon which information is presented to users.

screen device context. A logical description of a data destination that is a particular window on the screen. See also *device context*.

SCREEN\$. Character-device name reserved for the display screen.

scroll bar. A control window, horizontally or vertically aligned, that allows the user to scroll additional data into an associated panel area.

scrollable entry field. An entry field larger than the visible field.

scrollable selection field. A selection field that contains more choices than are visible.

scrolling. Moving a display image vertically or horizontally in a manner such that new data appears at one edge, as existing data disappears at the opposite edge.

secondary window. A type of window associated with the primary window in a dialog. A secondary window begins a secondary and parallel dialog that runs at the same time as the primary dialog.

sector. An addressable subdivision of a track used to record one block of program code or data on a disk or diskette.

segment. See *graphics segment*.

segment attributes. Attributes that apply to the segment as an entity, as opposed to the individual primitives within the segment. For example, the visibility or detectability of a segment.

segment chain. All segments in a graphics presentation space that are defined with the 'chained' attribute. Synonym for *picture chain*.

segment priority. The order in which segments are drawn.

segment store. An area in a normal graphics presentation space where retained graphics segments are stored.

select. To mark or choose an item. Note that *select* means to mark or type in a choice on the screen; *enter* means to send all selected choices to the computer for processing.

select button. The button on a pointing device, such as a mouse, that is pressed to select a menu choice. Also known as button 1.

selection cursor. A type of cursor used to indicate the choice or entry field users want to interact with. It is represented by highlighting the item that it is currently positioned on.

selection field. A field containing a list of choices from which the user can select one or more.

semaphore. An object used by multi-threaded applications for signalling purposes and for controlling access to serially reusable resources.

separator. See *panel body area separator*.

serial dialog box. See *modal dialog box*.

serialization. The consecutive ordering of items.

serialize. To ensure that one or more events occur in a specified sequence.

serially reusable resource (SRR). A logical resource or object that can be accessed by only one task at a time.

session. A routing mechanism for user interaction via the console; a complete environment that determines how an application runs and how users interact with the application. OS/2 can manage more than one session at a time, and more than one process can run in a session. Each session has its own set of environment variables that determine where OS/2 looks for dynamic-link libraries and other important files.

shadow box. The area on the screen that follows mouse movements and shows what shape the window will take if the mouse button is released.

shared data. Data that is used by two or more programs.

shared memory. Memory that is used by two or more programs.

shear. The tilt of graphics text when each character leans to the left or right while retaining a horizontal baseline.

shell. (1) A software interface between a user and the operating system of a computer. Shell programs interpret commands and user interactions on devices such as keyboards, pointing devices, and touch-sensitive screens, and communicate them to the operating system. (2) Software that allows a kernel program to run under different operating-system environments.

Shutdown. The procedure required before the computer is switched off to ensure that data is not lost.

sibling processes. Child processes that have the same parent process.

sibling windows. Child windows that have the same parent window.

slider box. An area on the scroll bar that indicates the size and position of the visible information in a panel area in relation to the information available. Also known as *thumb mark*.

source file. A file that contains source statements for items such as high-level language programs and data description specifications.

source statement. A statement written in a programming language.

specific dynamic-link module. A dynamic-link module created for the exclusive use of an application.

spline. A sequence of one or more Bézier curves.

spooler. A program that intercepts the data going to printer devices and writes it to disk. The data is printed or plotted when it is complete, and the required device is available. The spooler prevents output from different sources from being intermixed.

stack. A list constructed and maintained so that the next data element to be retrieved is the most recently stored. This method is characterized as last-in-first-out (LIFO).

standard window. A collection of window elements that form a panel. The standard window can include one or more of the following window elements: sizing borders, system menu icon, title bar, maximize/minimize/restore icons, action bar and pull-downs, scroll bars, and client area.

static control. The means by which the application presents descriptive information (for example, headings and descriptors) to the user. The user cannot change this information.

static storage. (1) A read/write storage unit in which data is retained in the absence of control signals. Static storage may use dynamic addressing or sensing circuits. (2) Storage other than *dynamic storage*.

style. See *window style*.

suballocation. The allocation of a part of one extent for occupancy by elements of a component other than the one occupying the remainder of the extent.

subdirectory. In an IBM personal computer, a file referred to in a root directory that contains the names of other files stored on the diskette or fixed disk.

swapping. (1) A process that interchanges the contents of an area of real storage with the contents of an area in auxiliary storage. (2) In a system with virtual storage, a paging technique that writes the active pages of a job to auxiliary storage and reads pages of another job from auxiliary storage into real storage. (3) The process of temporarily removing an active job from main storage, saving it on disk, and processing another job in the area of main storage formerly occupied by the first job.

switch. (1) An action that moves the input focus from one area to another. This can be within the same

window or from one window to another. (2) In a computer program, a conditional instruction and an indicator to be interrogated by that instruction. (3) A device or programming technique for making a selection, for example, a toggle, a conditional jump.

switch list. See *Task List*.

symbolic identifier. A text string that equates to an integer value in an include file, that is used to identify a programming object.

synchronous. Pertaining to events or operations that are predictable or occur at the same time. See also *asynchronous*.

System Menu. In PM, the pull-down in the top left corner of a window that allows it to be moved and sized with the keyboard.

system queue. This is the master queue for all pointer device or keyboard events.

Systems Application Architecture (SAA). A formal set of rules that enables applications to be run without modification in different computer environments.

T

tag. One or more characters attached to a set of data that defines the formatting or other characteristics of the set, including its definition.

Task List. In PM, the list of programs that are active. The list can be used to switch to a program and to stop programs.

template. An ASCII-text definition of an action bar and pull-down menu, held in a resource file, or as a data structure in program memory.

text. Characters or symbols.

text cursor. A symbol displayed in an entry field that indicates where typed input will appear.

text window. Also known as the VIO window.

text-windowed application. The environment in which the operating system performs advanced&hyphn.video input and output operations.

thread. A unit of execution within a process. It uses the resources of the process.

thumb mark. The portion of the scroll bar that describes the range and properties of the data that is currently visible in a window. Also known as a *slider box*.

tilde. A mark used to denote the character that is to be used as a mnemonic when selecting text items within a menu.

time slice. (1) An interval of time on the processing unit allocated for use in performing a task. After the interval has expired, processing-unit time is allocated to another task, so a task cannot monopolize processing-unit time beyond a fixed limit. (2) In systems with time sharing, a segment of time allocated to a terminal job.

title bar. The area at the top of a window that contains the window title. The title bar is highlighted when that window has the input focus. Contrast with *panel title*.

transaction. An exchange between a workstation and another device that accomplishes a particular action or result.

transform. (1) The action of modifying a picture by scaling, shearing, reflecting, rotating, or translating. (2) The object that performs or defines such a modification; also referred to as a *transformation*.

Tree. In PM, the window in the *File Manager* that shows the organization of drives and directories.

truncate. (1) To end a computational process in accordance with some rule. (2) To remove the beginning or ending elements of a string. (3) To drop data that cannot be printed or displayed in the line width specified or available. (4) To shorten a field or statement to a specified length.

U

unnamed pipe. A circular buffer, created in memory, used by related processes to communicate with one another. Contrast with *named pipe*.

update region. A system-provided area of dynamic storage containing one or more (not necessarily contiguous) rectangular areas of a window, that are visually invalid or incorrect, and therefore in need of repainting.

user interface. Hardware, software, or both that allows a user to interact with and perform operations on a system, program, or device.

User Shell. A component of OS/2 that uses a graphics-based, windowed interface to allow the user to manage applications and files installed and running under OS/2.

utility program. (1) A computer program in general support of computer processes; for example, a diagnostic program, a trace program, a sort program. (2) A program designed to perform an everyday task such as copying data from one storage device to another.

V

vector font. A set of symbols, each of which is created as a series of lines and curves. Synonymous with *outline font*. Contrast with *image font*.

VGA. Video graphics array.

viewing pipeline. The series of transformations applied to a graphic object to map the object to the device on which it is to be presented.

viewing window. Clipping boundary that defines the visible part of model space.

VIO. Video Input/Output.

virtual memory (VM). Addressable space that is apparent to the user as the processor storage space, but not having a fixed physical location.

virtual storage. Synonymous with *virtual memory*.

visible region. A window's presentation space, clipped to the boundary of the window and the boundaries of any overlying window.

volume. (1) A file-system driver that uses a block device driver for input and output operations to a local or remote device. (2) A portion of data, together with its data carrier, that can be handled conveniently as a unit.

W

wild-card character. The global file-name characters asterisk (*) and question mark (?).

window. A rectangular area of the screen with visible boundaries within which information is displayed. A window can be smaller than or the same size as the screen. Windows can appear to overlap on the screen.

window class. The grouping of windows whose processing needs conform to the services provided by one window procedure.

window coordinates. The means by which a window position or size is defined; measured in device units, or *pels*.

window procedure. Code that is activated in response to a message. The procedure controls the appearance and behavior of its associated windows.

window rectangle. The means by which the size and position of a window is described in relation to the desktop window.

window style. The set of properties that influence how events related to a particular window will be processed.

workstation. A display screen together with attachments such as a keyboard, a local copy device, or a tablet.

world coordinates. Application-convenient coordinates used for drawing graphics.

world-coordinate space. Coordinate space in which graphics are defined before transformations are applied.

WYSIWYG. What You See Is What You Get. A capability that enables text to be displayed on a screen in the same way it will be formatted on a printer.

Z

z-order. The order in which sibling windows are presented. The topmost sibling window obscures any portion of the siblings that it overlaps; the same effect occurs down through the order of lower sibling windows.

zooming. In graphics applications, the process of increasing or decreasing the size of picture.

Index

A

- ABB_* values 5-405, 5-463
- ACCEL A-1
- accelerator table
 - copy 8-37
 - create 8-44
 - destroy 8-98
 - load 8-234
 - query 8-291
 - set 8-439
 - translate 8-550
- ACCELTABLE A-1
- ACCELTABLE statement 32-9
- Access a DRAGINFO Structure 3-26
- Access Drag Information 3-4
- Add Atom 8-7
- Add Switch Entry 8-9
- Add Text to DDF Buffer 4-39
- additional metrics F-9
- addressing elements in arrays 1-5
- alarm sound 8-11
- Allocate DRAGINFO Structure 3-7
- Allocate DRAGTRANSFER Structures 3-9
- AM_* values 5-228, 5-401
- Animate Palette 5-8
- application-supplied functions 10-1
- Applications
 - Windowed PM 34-1
- Arabic text 5-435
- arc
 - create 5-199
 - full 5-148, 5-189
 - partial 5-188
 - query parameters 5-226
 - set current parameters 5-398
 - set default parameters 5-460
- Arc at a Given Position 33-3
- Arc at Current Position 33-3
- ARCPARAMS A-2
- AREABUNDLE A-2
- areas
 - begin construction 5-13
 - construction of interior 5-15
 - end construction 5-128
- arrays
 - addressing elements in 1-5
 - convert 5-53, 5-55
- ASCII 8-321, 8-459, 34-23
- ASCII MIXED code pages 34-23
- Associate 5-11
- Associate Help Instance 8-13
- ASSOCTABLE statement 32-10
- ATOM A-2
- attribute primitive type 5-404
- attribute primitive types 5-462
- attribute values
 - character 5-404, 5-462
 - image 5-405, 5-463
 - line 5-404, 5-462
 - marker 5-405, 5-463
 - pattern (area) 5-405, 5-463

attributes

- character-set 5-443
- color 5-453
- cosmetic line width 5-498
- foreground color mix 5-511
- geometric line width 5-500
- line type 5-495
- line width 5-498
- marker box 5-504
- marker set 5-506
- marker symbol 5-503
- pattern 5-522
- pattern set 5-526
- query mode 5-228
- restore saved 5-217
- segment 5-539
- set 5-404
- set default 5-462
- set line-end 5-491
- set line-join 5-493
- specify mode 5-401

ATTR_* values 5-304, 5-351, 5-488, 5-538

B

- background
 - query color 5-231, 5-232
 - query color-mixing mode 5-232
 - query mix 5-232
- BANDRECT A-2
- BA_* values 5-13
- BBO_* values 5-24, 5-113, 5-568
- BDS_* values 13-3
- Begin Area 5-13, 33-3
- Begin Definition List 4-2
- Begin Dragging Files 3-16
- Begin Element 5-17, 33-4
- Begin Image at Current Position 33-5
- Begin Image at Given Position 33-5
- Begin Paint 8-18
- Begin Path 5-19, 33-5
- Begin Window Enumeration 8-16
- Bezier Curve at Current Poition 33-6
- Bezier Curve at Given Position 33-6
- Bézier splines, create 5-215
- Bit Blt 5-23
- bit maps
 - color 5-25, 5-114, 5-569
 - copy rectangle of image data 5-23, 5-567
 - create 5-71
 - data D-1
 - delete 5-90
 - draw 8-118
 - example D-1
 - file format D-2
 - get system 8-194
 - information tables D-1
 - load 5-161
 - monochrome 5-25, 5-114, 5-569
 - query bits 5-233
 - query device formats 5-280
 - query dimension 5-236

character (continued)
 set mode 5-440
 set set 5-443
 set shear 5-445
 character attribute values 5-404, 5-462
 character definitions
 font F-3
 character direction
 Arabic text 5-435
 Chinese text 5-435
 Roman text 5-435
 character set 1-6
 Character String 5-34
 draw at current position 5-34
 draw at current position, with controls 5-39
 draw at specified position 5-36
 draw string at specified position, with controls 5-42
 Character String At 5-36
 Character String at Current Position 33-9
 Character String at Given Position 33-9
 Character String Extended at Current Position 33-10
 Character String Extended at Given Position 33-10
 Character String Move at Current Position 33-11
 Character String Move at Given Position 33-11
 Character String Position 5-39
 Character String Position At 5-42
 CHARBUNDLE A-11
 CHDIRN_* values 5-249, 5-435
 check box 13-1
 Check Menu Item 8-32
 Check Message Filter Hook 10-5
 CheckMsgFilterHook 10-5
 Chinese text 5-435
 CHS_* values 5-39, 5-42, 5-255, 5-257
 class 9-1
 CLASSDETAILS A-12
 CLASSINFO A-11
 clipboard 28-1
 messages 28-1
 query format information 8-310
 query viewer window 8-313
 set data 8-449
 clipboard messages 28-1
 clipping 5-528, G-1
 segment chains 5-122
 set path 5-448
 set region 5-451
 clipping boundary 5-486
 clipping region 8-150
 Close Clipboard 8-34
 Close Device Context 2-2
 Close Figure 5-45, 33-12
 Close Profile 6-2
 Close Segment 5-47
 closed figure 5-20
 CLR_* values 5-76, 5-231, 5-262, 5-338, 5-412, 5-453
 CMDSRC_* values 11-3, 12-27, 12-36, 12-63, 15-21
 CM_ALLOCDETAILFIELDINFO 24-22
 CM_ALLOCRECORD 24-23
 CM_ARRANGE 24-24
 CM_CLOSEEDIT 24-24
 CM_COLLAPSETREE 24-25
 CM_ERASERECORD 24-26
 CM_EXPANDTREE 24-26
 CM_FILTER 24-27
 CM_FREEDETAILFIELDINFO 24-28
 CM_FREERECORD 24-29
 CM_HORZSCROLLSPLITWINDOW 24-30
 CM_INSERTDETAILFIELDINFO 24-30
 CM_INSERTRECORD 24-31
 CM_INVALIDATEDDETAILFIELDINFO 24-33
 CM_INVALIDATERECORD 24-33
 CM_OPENEDIT 24-35
 CM_PAINTBACKGROUND 24-35
 CM_QUERYCNRINFO 24-36
 CM_QUERYDETAILFIELDINFO 24-37
 CM_QUERYDRAGIMAGE 24-38
 CM_QUERYRECORD 24-39
 CM_QUERYRECORDEMPHASIS 24-40
 CM_QUERYRECORDFROMRECT 24-41
 CM_QUERYRECORDINFO 24-42
 CM_QUERYRECORDRECT 24-43
 CM_QUERYVIEWPORTRECT 24-43
 CM_REMOVEDDETAILFIELDINFO 24-44
 CM_REMOVERECORD 24-45
 CM_SCROLLWINDOW 24-47
 CM_SEARCHSTRING 24-48
 CM_SETCNRINFO 24-49
 CM_SETRECORDEMPHASIS 24-50
 CM_SORTRECORD 24-51
 CM_* values 5-251, 5-427, 5-440
 CNRDRAGINFO A-12
 CNRDRAGINIT A-12
 CNRDRAWITEMINFO A-13
 CNREDITDATA A-14
 CNREDITDATA data structure A-13
 CNRINFO A-15
 CN_BEGINEDIT 24-8
 CN_COLLAPSETREE 24-9
 CN_CONTEXTMENU 24-9
 CN_DRAGAFTER 24-10
 CN_DRAGLEAVE 24-11
 CN_DRAGOVER 24-12
 CN_DROP 24-13
 CN_DROPHELP 24-14
 CN_EMPHASIS 24-15
 CN_ENDEDIT 24-15
 CN_ENTER 24-16
 CN_EXPANDTREE 24-17
 CN_HELP 24-17
 CN_INITDRAG 24-18
 CN_KILLFOCUS 24-19
 CN_QUERYDELTA 24-19
 CN_REALLOCPSZ 24-20
 CN_SCROLL 24-21
 CN_SETFOCUS 24-21
 CN_* values
 described 24-8
 code page
 query 8-314
 set 8-456
 Code Page Change Hook 10-7
 Code pages 34-1
 ASCII 34-11
 EBCDIC 34-16
 Font support 34-4
 OS/2 options for PM 34-3
 OS/2 support for multiple 34-4
 CodePageChangeHook 10-7
 COLOR A-20
 color palette 8-362
 color table G-1
 create 5-74
 color table default values 5-76

colors

- on monochrome devices 5-76
- query 5-262
- query data 5-264
- query foreground mix mode 5-324
- query index 5-266
- query nearest 5-327
- query real 5-343
- query RGB 5-349
- query system 8-362
- set 5-453
- set background 5-412
- set system values 8-494

Combine Region 5-49

combo box control data 19-1

combo box control window processing 19-1

Comment 5-51, 33-12

Compare Strings 8-35

constant names 1-1

constants

- button filtering 8-183

container control window processing

- data structures 24-3
- icon size, how determined A-17
- mini-icon size, how determined A-17
- notification codes 24-8
- notification messages 24-4
- purpose 24-1
- styles and selection types 24-2
- window messages 24-22
- window words 24-1

container views A-16

contents and format of dialog template 32-19

control classes 11-2

control codes

- Shift In (SI) 34-23
- Shift Out (SO) 34-23

control data 32-22

Control Formatting 4-35

control statements

- predefined 32-24

control window processing 11-2

CONVCONTEXT A-20

conventions

Convert 5-53

Convert with Matrix 5-55

coordinates

- dialog 32-19

coordinates for dialogs 32-19

Copy Accelerator Table 8-37

Copy Metafile 5-57

Copy Rectangle 8-39

Correlate Chain 5-59

Correlate From 5-63

Correlate Segment 5-67

cosmetic line width

- query 5-311

Counts Number of Items in Listbox 8-330

CPTTEXT A-21

Create a Paragraph in DDF Buffer 4-24

Create Accelerator Table 8-44

Create Atom Table 8-46

Create Bit Map 5-71

Create Cursor 8-48

Create Dialog 8-50

Create Frame Controls 8-52

Create Help Instance 8-54

Create Help Table 8-56

Create Logical Color Table 5-74

Create Logical Font 5-78

Create Menu 8-58

Create Message Queue 8-60

Create Palette 5-81

Create Pointer 8-64

Create Pointer Indirect 8-66

Create Presentation Space 5-84

Create Region 5-88

Create Standard Window 8-68

Create String Handle 3-5

Create Switch Entry 8-72

Create Window 8-74

Create Workplace Object 8-62

CREATESTRUCT A-21

CREA_* values 5-195

CRGN_* values 5-49

CS_* values

- window class styles 12-1

CTAB_* values 5-195

CTIME A-22

current position

- move 5-173
- query 5-269
- set to specified point 5-458

cursor

- create 8-48
- destroy 8-101
- hide 8-518
- query information 8-316
- show 8-518

CURSORINFO A-22

CURSOR_* values 8-48

CVR_* values 12-23

CVTC_* values 5-53

CV_* values

- CNRINFO structure A-16
- SEARCHSTRING structure A-115
- view styles A-17

D

data

- bit map D-1
- get 5-150
- put 5-223

data area in a dialog template 32-22

data format

- image F-7
- outline F-8

data types A-1

- graphics orders 33-1
- implicit pointer 1-5
- storage mapping 1-6

DBCS 8-285

DBCS support 34-23

- character-encoding schemes 34-23

DBM_* values 8-118

DB_* values 8-121

DCTL_* values 5-282, 5-474

DC_* values A-32

DDEF_* values 5-195

DDEINIT A-23

DDESTRUCT A-23

DDE_* values 30-1, 30-2, 30-3, A-23

DdfBeginList 4-2

- DdfBitmap 4-5
- DdfEndList 4-8
- DdfHyperText 4-10
- DdfInform 4-13
- DdfInitialize 4-15
- DdfListItem 4-18
- DdfMetafile 4-21
- DdfPara 4-24
- DdfSetColor 4-26
- DdfSetFont 4-29
- DdfSetFontStyle 4-32
- DdfSetFormat 4-35
- DdfSetTextAlign 4-37
- DdfText 4-39
- default colors 13-2, 14-2, 15-3, 16-1, 17-3, 19-2, 20-2, 22-2, 23-1
- Default Dialog Procedure 8-85
- default dialog processing 12-70
- default graphics character box
 - query 5-275
- default message processing 12-1
- default view matrix
 - query 5-273
- Default Window Procedure 8-89
- default window processing 11-1
- DEFAULTICON keyword 32-11
- Define Hypertext Link 4-10
- Define Inform Link 4-13
- Define Text Alignment 4-37
- Delete Atom 8-91
- Delete Bit Map 5-90
- Delete DRAGINFO String Handles 3-10
- Delete Element 5-92
- Delete Element Range 5-94
- Delete Elements Between Labels 5-96
- Delete Library 8-95
- Delete Listbox Item 8-93
- Delete Metafile 5-98
- Delete Palette 5-100
- Delete Procedure 8-96
- Delete Segment 5-102
- Delete Segments 5-104
- Delete Set Identifier 5-106
- Delete String Handle 3-11
- DELETENOTIFY A-24
- Deregister Workplace Object Class 8-97
- DESKTOP A-24
- Destroy Accelerator Table 8-98
- Destroy Atom Table 8-99
- Destroy Cursor 8-101
- Destroy Help Instance 8-102
- Destroy Message Queue 8-104
- Destroy Pointer 8-107
- Destroy Presentation Space 5-108
- Destroy Region 5-110
- Destroy Window 8-109
- Destroy Window Hook 10-8
- Destroy Workplace Object 8-106
- DestroyWindowHook 10-8
- detectability attribute for segments
 - modify (GpiSetSegmentAttrs) 5-539
- DevCloseDC 2-2
- DevEscape 2-4
- DEVESC_* values 2-4, 2-5
- device characteristics
 - query 2-15
- device context
 - clear output display 5-136
 - close 2-2
 - create 2-9
 - open 2-9
 - open for a window 8-273
 - screen 8-128
- DevOpenDC 2-9
- DEVOPENSTRUC A-25
- DevPostDeviceModes 2-12
- DevQueryCaps 2-15
- DevQueryDeviceNames 2-21
- DevQueryHardcopyCaps 2-24
- DEV_* values 2-2, 2-10
- DFORM_* values 5-150, 5-223
- dialog
 - create 8-50
 - default procedure 8-85
 - dismiss 8-111
 - enumerate item 8-145
 - load 8-236
 - process modal 8-287
 - query item short 8-321
 - send message to item 8-435
 - set item short 8-459
- dialog item
 - query text 8-323
 - query text length 8-325
 - set text 8-461
- dialog points
 - map 8-259
- Dialog Procedure 10-2
- dialog processing 12-70
 - default 12-70
 - language support 12-83
- dialog template
 - data-area information 32-22
 - format and contents 32-19
 - header information 32-20
 - item information 32-21
- dialog window
 - destroy modal 8-111
 - hide modeless 8-111
- DialogProc 10-2
- dialogs
 - define procedure 10-2
- Direct Manipulation for Files 3-2
- direct manipulation messages 29-1
- directives 32-4
- Dismiss Dialog 8-111
- Dispatch Message 8-113
- dithered colors 5-327
- dithering 5-327, 8-494
- DLGC_* values 12-72
- DLGTEMPLATE A-27
- DLGTEMPLATE statement 32-16
- DLGTITEM A-27
- DM_DISCARDOBJECT 29-1
- DM_DRAGERROR 29-2
- DM_DRAGFILECOMPLETE 29-2
- DM_DRAGLEAVE 29-3
- DM_DRAGOVER 29-4
- DM_DRAGOVERNOTIFY 29-5
- DM_DROP 29-6
- DM_DROPHELP 29-7
- DM_EMPHASIZETARGET 29-7
- DM_ENDCONVERSATION 29-8

- DM_FILERENDERED 29-9
- DM_PRINTOBJECT 29-9
- DM_RENDER 29-10
- DM_RENDERCOMPLETE 29-11
- DM_RENDERFILE 29-12
- DM_RENDERERPREPARE 29-13
- DM_* values 5-284, 5-477
- double-byte character set 1-6
- double-byte character sets 34-23
- Down cursor key 8-547
- DO_* values
 - DRAGINFO data structure A-29
 - DRAGITEM data structure A-32
- DPC errors 5-2
- DPDM_* values 2-13
- DP_* values 8-124
- Drag 3-12
- drag information
 - access 3-4
- drag messages 29-1
- DRAGIMAGE A-28
- DRAGINFO A-29
- DRAGITEM A-30
- DRAGTRANSFER A-32
- Draw Bit Map 8-118
- Draw Bits 5-112
- Draw Border 8-121
- Draw Chain 5-117
- Draw Dynamics 5-119
- Draw From 5-121
- draw mode 5-47
- Draw Pointer 8-124
- Draw Polygons 5-207
- Draw Segment 5-123
- Draw Text 8-126
- Draw Tracking Rectangle 8-546
- draw-and-retain mode 5-47
- drawing mode
 - draw 5-126, 5-474, 5-478, 5-558
 - draw-and-retain 5-126, 5-287, 5-474, 5-478, 5-558
 - query 5-284
 - retain 5-126, 5-252, 5-287, 5-478, 5-558
 - set 5-477
- drawing orders 33-1
- drawing process check errors 5-2
- DRF_* values A-31
- DrgAcceptDroppedFiles 3-2
- DrgAccessDraginfo 3-4
- DrgAddStrHandle 3-5
- DrgAllocDraginfo 3-7
- DrgAllocDragtransfer 3-9
- DrgDeleteDraginfoStrHandles 3-10
- DrgDeleteStrHandle 3-11
- DrgDrag 3-12
- DrgDragFiles 3-16
- DrgFreeDraginfo 3-19
- DrgFreeDragtransfer 3-21
- DrgGetPS 3-22
- DrgPostTransferMsg 3-24
- DrgPushDraginfo 3-26
- DrgQueryDragitem 3-28
- DrgQueryDragitemCount 3-30
- DrgQueryDragitemPtr 3-31
- DrgQueryNativeRMF 3-32
- DrgQueryNativeRMFLen 3-34
- DrgQueryStrName 3-36
- DrgQueryStrNameLen 3-38

- DrgQueryTrueType 3-40
- DrgQueryTrueTypeLen 3-42
- DrgReleasePS 3-44
- DrgSendTransferMsg 3-45
- DrgSetDragImage 3-48
- DrgSetDragitem 3-50
- DrgSetDragPointer 3-53
- DrgVerifyNativeRMF 3-55
- DrgVerifyRMF 3-57
- DrgVerifyTrueType 3-59
- DrgVerifyType 3-61
- DrgVerifyTypeSet 3-63
- DRG_* values A-29
- DRIVDATA A-33
- DRIVPROPS A-34
- DRM_* values A-31
- DRO_* values 5-28, 5-148
- DRT_* values A-30
- DTYP_* values 8-408
- DT_* values 8-127, 22-1
- Dynamic Data Exchange Initiate (NLS) 8-78
- dynamic data exchange messages 30-1
- Dynamic Data Exchange Post Message (NLS) 8-80
- Dynamic Data Exchange Respond (NLS) 8-83

E

- EBCDIC MIXED code pages 34-23
- edit mode
 - query 5-285
 - set 5-480
- EDI_* values 8-145
- EGA 2-19
- Element 5-125
 - end 5-130
 - query 5-286
- elements
 - delete 5-92
 - delete between labels 5-96
 - delete between range 5-94
 - offset pointer 5-177
 - query pointer 5-288
 - query type 5-290
 - set pointer at label 5-484
- Empty Clipboard 8-130
- EM_CLEAR 14-4
- EM_COPY 14-4
- EM_CUT 14-5
- EM_PASTE 14-5
- EM_QUERYCHANGED 14-6
- EM_QUERYFIRSTCHAR 14-7
- EM_QUERYREADONLY 14-7
- EM_QUERYSEL 14-8
- EM_SETFIRSTCHAR 14-8
- EM_SETINSERTMODE 14-9
- EM_SETREADONLY 14-10
- EM_SETSEL 14-10
- EM_SETTEXTLIMIT 14-11
- Enable Control of Button Id 8-131
- Enable Menu Item 8-132
- Enable Physical Input 8-134
- Enable Window Update 8-137
- encapsulation 9-1
- End Area 5-128, 33-13
- End Definition List 4-8
- End Element 5-130, 33-13
- End Image 33-13

- End of Symbol Definition 33-14
- End Paint 8-141
- End Path 5-132, 33-14
- End Prolog 33-14
- End Window Enumeration 8-139
- ENDFONT structure F-1
- Enter key 8-547
- entry field control data 14-2
- entry field control window processing 14-1
- ENTRYFDATA A-34
- Enumerate Clipboard Formats 8-143
- Enumerate Dialog Item 8-145
- Enumerate Object Classes 8-147
- EN_* values 14-3, 18-3
- EQRGN_* values 5-134
- Equal Rectangle 8-148
- Equal Region 5-134
- Erase 5-136
- ERRINFO A-35
- Error Segment Data 5-138
- error severities 1-2
- error state
 - get last one 8-178
- error-information block 8-165
- ERRORID A-35
- errors
 - codes B-1
 - drawing process check 5-2
 - explanations C-1
 - get information 8-175
 - severities of 1-2
- Esc key 8-547
- Escape 2-4, 33-15
- ESCSETMODE A-35
- ES_* dbcsvals 14-2
- ES_* values 14-1
- Exclude Clip Rectangle 5-140
- Exclude Update Region 8-150
- Extended Escape 33-15

F

- FACENAMEDESC A-35
- FATTRS A-36
- FATTR_FONTUSE_* values A-38
- FATTR_SEL_* values A-37
- FATTR_TYPE_* values A-38
- FCF_* frame styles 8-424
- FCF_* values 15-1
- FC_* values 8-160
- FDATE A-38
- FDM_ERROR 12-73
- FDM_FILTER 12-74
- FDM_VALIDATE 12-74
- FDS_* values A-42
- FFDESCS A-39
- FFDESCS2 A-39
- FF_* indicators 8-400
- FF_* values 5-144
- FID_* values 15-1, 23-1
- FIELDINFO A-39
- FIELDINFOINSERT A-41
- FIELDINFOINSERT data structure A-41
- file dialog 12-73
- file format
- file formats
 - bit maps D-2

- file formats (*continued*)
 - icon file D-2
 - pointer D-2
- FILEDLG A-42
- FILEFINDBUF4 A-46
- Fill Path 5-142, 33-16
- Fill Rectangle 8-154
- Fillet at Current Position 33-16
- Fillet at Given Position 33-16
- Find Atom 8-156
- Find Word Hook 10-9
- FindWordHook 10-9
- FIXED A-46
- FI_* values 15-18
- Flash Window 8-158
- flashing
 - start 8-158
 - stop 8-158
- flipping bits 8-211
- Flood Fill 5-144
- FM_* values 5-324, 5-510
- FNTF_* values A-49
- FNTM_FACENAMECHANGED 12-76
- FNTM_FILTERLIST 12-77
- FNTM_POINTSIZEXCHANGED 12-78
- FNTM_STYLECHANGED 12-78
- FNTM_UPDATEPREVIEW 12-79
- FNTS_* values A-48
- FOCAMETRICS structure F-2
- focus
 - change window 8-160
 - query 8-327
 - set window 8-464
- FOLDERDATA A-46
- font character definitions F-3
- font definition header F-4
- font dialog 12-75
- font directory F-11
- font metrics F-1
- font-file format F-1
- FONTDEFINITIONHEADER structure F-4
- FONTDLG A-47
- FONTMETRICS A-52
- fonts
 - create logical definition 5-78
 - definition of terms F-12
 - Japanese 34-23
 - load 5-163
 - load public 5-167
 - outline 5-427, 5-430, 5-433, 5-438, 5-445
 - query 5-299
 - query action 5-294
 - query face string 5-292
 - query logical 5-315
 - query metrics 5-297
 - query number of local identifiers 5-329
 - query set identifiers 5-359
 - query width table 5-372
 - raster 5-427, 5-430, 5-433, 5-438, 5-445, 5-522
 - unload 5-563
 - unload public 5-565
- fonts supplied with OS/2 E-1
- FONTSIGNATURE structure F-1
- FONT_* values 5-78
- format
 - font-file F-1
- format and contents of dialog template 32-19

- FPATH_* values 5-142, 5-191
- frame control data 15-3
- frame control window processing 15-1
- Frame Region 5-146
- FRAMECDATA A-60
- Free DRAGINFO Structure 3-19
- Free DRAGTRANSFER Storage 3-21
- Free Error Information 8-165
- Free File Icon 8-168
- Free Standard File Dialog File List 8-166
- FS_* values 15-3
- FTIME A-61
- Full Arc 5-148
 - create 5-148
- Full Arc at Current Position 33-17
- Full Arc at Given Position 33-17
- function descriptions
 - conventions used 1-1
- functions
 - supplied by applications 10-1

G

- GARC 33-3
- GBAR 33-3
- GBBLT 33-7
- GBEL 33-4
- GBEZ 33-6
- GBIMG 33-5
- GBIT1 33-1
- GBIT16 33-1
- GBIT2 33-1
- GBIT32 33-1
- GBIT4 33-1
- GBIT5 33-1
- GBIT6 33-1
- GBIT7 33-1
- GBIT8 33-1
- GBOX 33-8
- GBPTH 33-5
- GCALLS 33-9
- GCARC 33-3
- GCBEZ 33-6
- GCBIMG 33-5
- GCBBOX 33-8
- GCCHST 33-9
- GCCHSTE 33-10
- GCCHSTM 33-11
- GCFARC 33-17
- GCFLT 33-16
- GCHAR 33-1
- GCHST 33-9
- GCHSTE 33-10
- GCHSTM 33-11
- GCLFIG 33-12
- GCLINE 33-18
- GCMRK 33-18
- GCOMT 33-12
- GCPARC 33-20
- GCRLINE 33-22
- GCSFLT 33-50
- GDELPOINT 33-1
- GEAR 33-13
- GEEL 33-13
- GEESCP 33-15
- GEIMG 33-13
- general window styles 12-1

- geometric line width 5-312
- GEPROL 33-14
- GEPH 33-14
- GESCP 33-15
- GESD 33-14
- Get Clipped Presentation Space 8-169
- Get Current Time 8-171
- Get Data 5-150
- Get Dialog Message 8-172
- Get Drag Presentation Space 3-22
- Get Dragged Object Count 3-30
- Get DRAGITEM Structure 3-28
- Get Error Information 8-175
- Get Format of a Dragged Object 3-32
- Get Key State 8-176
- Get Last Error 8-178
- Get Maximum Position 8-179
- Get Message 8-183
- Get Minimum Position 8-181
- Get Multiple Windows From Identities 8-266
- Get Next Window 8-186
- Get Physical Key State 8-188
- Get Pointer to DRAGITEM Structure 3-31
- Get Presentation Space 8-190
- Get Screen Presentation Space 8-192
- Get String Contents 3-36
- Get String Length 3-38
- Get String Length for Native RMF of Dragged Object 3-34
- Get String Length for True Type of Dragged Object 3-42
- Get System Bit Map 8-194
- Get True Type of Dragged Object 3-40
- GFARC 33-17
- GFIXED 33-2
- GFIXEDS 33-2
- GFLT 33-16
- GFPATH 33-16
- GHBITMAP 33-2
- GIMD 33-17
- GINDATT 33-2
- GINDEX3 33-2
- GLBL 33-18
- GLENGTH1 33-2
- GLENGTH2 33-2
- GLINE 33-18
- GLONG 33-2
- GMPATH 33-19
- GMRK 33-18
- GNOP1 33-19
- GOPATH 33-19
- GPARC 33-20
- GpiAnimatePalette 5-8
- GpiAssociate 5-11
- GpiBeginArea 5-13
- GpiBeginElement 5-17
- GpiBeginPath 5-19
- GpiBitBlt 5-23
- GpiBox 5-28
- GpiCallSegmentMatrix 5-31
- GpiCharString 5-34
- GpiCharStringAt 5-36
- GpiCharStringPos 5-39
- GpiCharStringPosAt 5-42
- GpiCloseFigure 5-45
- GpiCloseSegment 5-47
- GpiCombineRegion 5-49
- GpiComment 5-51

GpiConvert 5-53
 GpiConvertWithMatrix 5-55
 GpiCopyMetaFile 5-57
 GpiCorrelateChain 5-59
 GpiCorrelateFrom 5-63
 GpiCorrelateSegment 5-67
 GpiCreateBitmap 5-71
 GpiCreateLogColorTable 5-74
 GpiCreateLogFont 5-78
 GpiCreatePalette 5-81
 GpiCreatePS 5-84
 GpiCreateRegion 5-88
 GpiDeleteBitmap 5-90
 GpiDeleteElement 5-92
 GpiDeleteElementRange 5-94
 GpiDeleteElementsBetweenLabels 5-96
 GpiDeleteMetaFile 5-98
 GpiDeletePalette 5-100
 GpiDeleteSegment 5-102
 GpiDeleteSegments 5-104
 GpiDeleteSetId 5-106
 GpiDestroyPS 5-108
 GpiDestroyRegion 5-110
 GpiDrawBits 5-112
 GpiDrawChain 5-117
 GpiDrawDynamics 5-119
 GpiDrawFrom 5-121
 GpiDrawSegment 5-123
 GpiElement 5-125
 GpiEndArea 5-128
 GpiEndElement 5-130
 GpiEndPath 5-132
 GpiEqualRegion 5-134
 GpiErase 5-136
 GpiErrorSegmentData 5-138
 GpiExcludeClipRectangle 5-140
 GPIE_* values 5-138
 GpiFillPath 5-142
 GpiFloodFill 5-144
 GpiFrameRegion 5-146
 GpiFullArc 5-148
 GPIF_* values 5-533
 GpiGetData 5-150
 GpiImage 5-153
 GpiIntersectClipRectangle 5-155
 GpiLabel 5-157
 GpiLine 5-159
 GpiLoadBitmap 5-161
 GpiLoadFonts 5-163
 GpiLoadMetaFile 5-165
 GpiLoadPublicFonts 5-167
 GpiMarker 5-168
 GpiModifyPath 5-170
 GpiMove 5-173
 GpiOffsetClipRegion 5-175
 GpiOffsetElementPointer 5-177
 GpiOffsetRegion 5-179
 GpiOpenSegment 5-181
 GpiOutlinePath 5-184
 GpiPaintRegion 5-186
 GpiPartialArc 5-188
 GpiPathToRegion 5-191
 GpiPlayMetaFile 5-193
 GpiPointArc 5-199
 GpiPolyFillet 5-201
 GpiPolyFilletSharp 5-204
 GpiPolygons 5-207
 GpiPolyLine 5-209
 GpiPolyLineDisjoint 5-211
 GpiPolyMarker 5-213
 GpiPolySpline 5-215
 GpiPop 5-217
 GpiPtInRegion 5-219
 GpiPtVisible 5-221
 GpiPutData 5-223
 GpiQueryArcParams 5-226
 GpiQueryAttrMode 5-228
 GpiQueryAttrs 5-229
 GpiQueryBackColor 5-231
 GpiQueryBackMix 5-232
 GpiQueryBitmapBits 5-233
 GpiQueryBitmapDimension 5-236
 GpiQueryBitmapHandle 5-239
 GpiQueryBitmapInfoHeader 5-237
 GpiQueryBitmapParameters 5-240
 GpiQueryBoundaryData 5-242
 GpiQueryCharAngle 5-244
 GpiQueryCharBox 5-246
 GpiQueryCharBreakExtra 5-248
 GpiQueryCharDirection 5-249
 GpiQueryCharExtra 5-250
 GpiQueryCharMode 5-251
 GpiQueryCharSet 5-252
 GpiQueryCharShear 5-253
 GpiQueryCharStringPos 5-255
 GpiQueryCharStringPosAt 5-257
 GpiQueryClipBox 5-259
 GpiQueryClipRegion 5-261
 GpiQueryColor 5-262
 GpiQueryColorData 5-264
 GpiQueryColorIndex 5-266
 GpiQueryCp 5-268
 GpiQueryCurrentPosition 5-269
 GpiQueryDefArcParams 5-270
 GpiQueryDefAttrs 5-271
 GpiQueryDefaultViewMatrix 5-273
 GpiQueryDefCharBox 5-275
 GpiQueryDefTag 5-277
 GpiQueryDefViewingLimits 5-278
 GpiQueryDevice 5-279
 GpiQueryDeviceBitmapFormats 5-280
 GpiQueryDrawControl 5-282
 GpiQueryDrawingMode 5-284
 GpiQueryEditMode 5-285
 GpiQueryElement 5-286
 GpiQueryElementPointer 5-288
 GpiQueryElementType 5-290
 GpiQueryFaceString 5-292
 GpiQueryFontAction 5-294
 GpiQueryFontFileDescriptions 5-295
 GpiQueryFontMetrics 5-297
 GpiQueryFonts 5-299
 GpiQueryFullFontFileDescriptions 5-301
 GpiQueryGraphicsField 5-303
 GpiQueryInitialSegmentAttrs 5-304
 GpiQueryKerningPairs 5-306
 GpiQueryLineEnd 5-308
 GpiQueryLineJoin 5-309
 GpiQueryLineType 5-310
 GpiQueryLineWidth 5-311
 GpiQueryLineWidthGeom 5-312
 GpiQueryLogColorTable 5-313
 GpiQueryLogicalFont 5-315
 GpiQueryMarker 5-317

GPSIA 33-35
 GPSICOL 33-34
 GPSLE 33-36
 GPSLJ 33-36
 GPSLT 33-37
 GPSLW 33-38
 GPSMC 33-39
 GPSMP 33-40
 GPSMS 33-40
 GPSMT 33-41
 GPSMX 33-41
 GPSPIK 33-45
 GPSPRP 33-43
 GPSPS 33-44
 GPSPT 33-44
 GPSSLW 33-46
 GPSTA 33-47
 GPSTM 33-42
 GPSVW 33-48
 GRADIENTL A-61
 graphics
 orders 33-1
 query field 5-303
 set field 5-486
 graphics orders
 data types 33-1
 GREAL 33-2
 GRES_* values 5-382
 GRLINE 33-22
 GROF 33-2
 GROFUF 33-2
 GROL 33-2
 GROSOL 33-2
 GROUFS 33-2
 GROUL 33-2
 GSAP 33-23
 GSBCOL 33-23
 GSBICOL 33-24
 GSBMX 33-25
 GSCA 33-26
 GSCBE 33-26
 GSCC 33-27
 GSCD 33-28
 GSCE 33-28
 GSCH 33-30
 GSCOL 33-31
 GSCP 33-32
 GSCPTH 33-31
 GSCR 33-29
 GSCS 33-30
 GSECOL 33-32
 GSFLT 33-50
 GSFLW 33-33
 GSGCH 33-22
 GSHORT 33-2
 GSHORT370 33-2
 GSIA 33-35
 GSICOL 33-34
 GSLE 33-36
 GSLJ 33-36
 GSLT 33-37
 GSLW 33-38
 GSMC 33-39
 GSMP 33-40
 GSMS 33-40
 GSMT 33-41
 GSMX 33-41

GSPIK 33-45
 GSPRP 33-43
 GSPS 33-44
 GSPT 33-44
 GSSB 33-45
 GSSLW 33-46
 GSTA 33-47
 GSTM 33-42
 GSTR 33-2
 GSTV 33-48
 GSVW 33-48
 GUCHAR 33-2
 GUFIXEDS 33-3
 GULONG 33-3
 GULONG370 33-3
 GUNDF 33-3
 GUNDF1 33-3
 GUSHORT 33-3
 GUSHORT370 33-3

H

HAB A-61
 HACCEL A-61
 HAPP A-61
 HATOMTBL A-61
 HBITMAP A-61
 HCAPS_* values A-62
 HCINFO A-61
 HDC A-62
 HDDF A-62
 header 32-20
 header files 1-3
 Help Hook 10-10
 help manager messages 31-1
 helper macros 1-3
 HelpHook 10-10
 HELPINIT A-62
 HELPTABLE A-63
 HENUM A-64
 HEV A-64
 HFILE A-64
 HFIND A-64
 HFM_* values 10-10
 HIGHER_* values 5-355, 5-541
 highlight attribute for segments
 modify (GpiSetSegmentAttrs) 5-539
 HINI A-64
 HK_* values 8-466
 HLIB A-64
 HMERR_* error constants 31-4
 HMF A-64
 HMODULE A-64
 HMQ A-64
 HMQ_* values 8-418
 HMTX A-64
 HMUX A-64
 HM_ACTIONBAR_COMMAND 31-1
 HM_CONTROL 31-1
 HM_CREATE_HELP_TABLE 31-2
 HM_DISMISS_WINDOW 31-2
 HM_DISPLAY_HELP 31-3
 HM_ERROR 31-4
 HM_EXT_HELP 31-5
 HM_EXT_HELP_UNDEFINED 31-6
 HM_GENERAL_HELP 31-6
 HM_GENERAL_HELP_UNDEFINED 31-7

- HM_HELPSUBITEM_NOT_FOUND 31-8
- HM_HELP_CONTENTS 31-7
- HM_HELP_INDEX 31-8
- HM_INFORM 31-9
- HM_INVALIDATE_DDF_DATA 31-10
- HM_KEYS_HELP 31-10
- HM_LOAD_HELP_TABLE 31-11
- HM_NOTIFY 31-12
- HM_QUERY 31-13
- HM_QUERY_DDF_DATA 31-14
- HM_QUERY_KEYS_HELP 31-14
- HM_REPLACE_HELP_FOR_HELP 31-15
- HM_REPLACE_USING_HELP 31-15
- HM_SET_ACTIVE_WINDOW 31-16
- HM_SET_COVERPAGE_SIZE 31-17
- HM_SET_HELP_LIBRARY_NAME 31-17
- HM_SET_HELP_WINDOW_TITLE 31-18
- HM_SET_OBJCOM_WINDOW 31-18
- HM_SET_SHOW_PANEL_ID 31-19
- HM_SET_USERDATA 31-19
- HM_TUTORIAL 31-20
- HM_UPDATE_OBJCOM_WINDOW_CHAIN 31-21
- HOBJECT A-64
- hook
 - change code page 10-7
 - find word 10-9
 - help requests 10-10
 - input 10-8, 10-13
 - message filter 10-20
 - release 8-418
 - send message 10-23
 - set 8-466
- hooks 10-1
- HPAL A-64
- HPOINTER A-64
- HPROC A-64
- HPROGARRAY A-64
- HPROGRAM A-65
- HPS A-65
- HRGN A-65
- HRGN_* values 5-451
- HSEM A-65
- HSPL A-65
- HSTR A-65
- HSVWP A-65
- HSWITCH A-65
- HT_* values 12-37
- HWND A-65
- HWND_* values 8-11, 8-50, 8-52, 8-58, 8-115, 8-236, 8-244, 8-260, 8-362, 8-506

I

- IBB_* values 5-405, 5-463
- icon
 - destroy 8-107
- icon file format D-2
- icon size, how determined A-17
- ICONINFO A-65
- IconPos A-66
- Image 5-153
 - draw 5-153
- image attribute values 5-405, 5-463
- Image Data 33-17
- IMAGEBUNDLE A-66
- Implicit Pointer 1-1
- implicit pointer data types 1-5

- In Send Message 8-201
- Inflate Rectangle 8-197
- information tables
 - bit map D-1
- inheritance 9-1
- initialization file H-1
- Initialize 8-199
- Initialize DDF Area 4-15
- initialize Presentation Interface 8-199
- Input Hook 10-13
- InputHook 10-13
- Insert List Item 4-18
- Insert Listbox Item 8-203
- interchange file format G-1
- Intersect Clip Rectangle 5-155
- Intersect Rectangle 8-205
- Invalidate Rectangle 8-207
- Invalidate Region 8-209
- Invert Rectangle 8-211
- IPT A-66
- Is Child 8-213
- Is Control Enabled 8-214
- Is Menu Item Checked 8-216
- Is Menu Item Enabled 8-218
- Is Menu Item Valid 8-220
- Is Physical Input Enabled 8-222
- Is Rectangle Empty 8-223
- Is Thread Active 8-224
- Is Window 8-226
- items in a dialog template 32-21

J

- Japanese fonts 34-23
- Journal Playback Hook 10-14
- Journal Record Hook 10-15
- JournalPlaybackHook 10-14
- JournalRecordHook 10-15
- JRN_* values 12-39

K

- kanji 34-23
- KC_* values 12-24
- kerning A-60
 - device support 2-18
 - enable A-38
 - number of pairs A-60
 - query pairs 5-306
- kerning pair table F-8
- KERNINGPAIRS A-66
- KERNINGPAIRS data structure A-66
- Keyboard control codes 12-24
- keyboard resources 32-18
- keyboard statements
 - keyboard 32-18
- KS_* values 8-176, 8-188

L

- Label 5-157, 33-18
 - generate element for 5-157
- language support dialog processing 12-83
- language support window processing 12-80
- LBB_* values 5-404, 5-462
- LCIDT_* values 5-359

- LCID_* values 5-252, 5-320, 5-337, 5-443, 5-506, 5-526
- LCOLF_* values 5-74, 5-264, 8-494
- LCOLOPT_* 5-349
- LCOLOPT_* values 5-313, 5-333, 5-343
- LCOL_* options 8-494
- LCOL_* values 5-74, 5-264
- LC_* values 5-194
- Left cursor key 8-547
- LHANDLE A-66
- Line 5-159
 - draw 5-159
 - query cosmetic width 5-311
 - query end 5-308
 - query geometric width 5-312
 - query join 5-309
 - query type 5-310
 - query width 5-311
 - set cosmetic width 5-498
 - set end 5-491
 - set geometric width 5-500
 - set join 5-493
 - set type 5-495
 - set width 5-498
- Line at Current Position 33-18
- Line at Given Position 33-18
- line attribute values 5-404, 5-462
- LINEBUNDLE A-66
- LINEEND_* values 5-308, 5-491
- LINEJOIN_* values 5-309, 5-493
- LINETYPE_* values 5-310, 5-495
- LINEWIDTHGEOM_* values 5-312
- LINEWIDTH_* values 5-311, 5-498
- list box control data 16-1
- list box control styles 16-1
- list box control window processing 16-1
- LIT_* values 16-6
- LM_DELETEALL 16-5
- LM_DELETEITEM 16-5
- LM_INSERTITEM 16-6
- LM_QUERYITEMCOUNT 16-7
- LM_QUERYITEMHANDLE 16-7
- LM_QUERYITEMTEXT 16-8
- LM_QUERYITEMTEXTLENGTH 16-9
- LM_QUERYSELECTION 16-9
- LM_QUERYTOPINDEX 16-10
- LM_SEARCHSTRING 16-11
- LM_SELECTITEM 16-12
- LM_SETITEMHANDLE 16-12
- LM_SETITEMHEIGHT 16-13
- LM_SETITEMTEXT 16-14
- LM_SETTOPINDEX 16-14
- LN_* values 16-2
- Load Accelerator Table 8-234
- Load and Process Modal Dialog 8-115
- Load Bit Map 5-161
- Load Dialog 8-236
- Load File Icon 8-239
- Load Fonts 5-163
- Load Help Table 8-241
- Load Library 8-243
- Load Menu 8-244
- Load Message 8-246
- Load Metafile 5-165
- Load Pointer 8-248
- Load Procedure 8-250
- Load Public Fonts 5-167
- Load String 8-251

- load type options 5-193
- Loader Hook 10-16
- LoaderHook 10-16
- LOADOPTION 32-2
- local identifier options 5-193
- Lock Visible Regions 8-253
- Lock Window Update 8-255
- logical color table
 - create 5-74
- logical font
 - delete 5-106
- LONG A-67
- LOWER_* values 5-355, 5-541
- LSS_* values 16-11
- LS_* values 16-1
- LT_* values 5-193

M

- Make Points 8-257
- Make Rectangle 8-258
- Map Dialog Points 8-259
- Map Window Points 8-260
- Marker 5-168
 - draw a series of 5-213
 - draw with center at specified position 5-168
 - query 5-317
 - query box 5-318
 - query set 5-320
 - query symbol 5-317
 - set 5-502
 - set box 5-504
 - set set 5-506
- Marker at Current Position 33-18
- Marker at Given Position 33-18
- marker attribute values 5-405, 5-463
- MARKERBUNDLE A-67
- MARKSYM_* values 5-317, 5-502
- MATRIXLF A-68
- MBB_* values 5-463
- MBID_* values 8-264
- MB_* values 8-262, 8-263
- MEMOPTION 32-2
- memory
 - release 8-165
- MEMORYITEM A-68
- menu control styles 17-1
- menu control window processing 17-1
- menu item attributes 17-2
- menu item styles 17-2
- MENU statement 32-11
- MENUITEM A-68
- menus
 - create 8-58
 - create window 8-58
 - load 8-244
 - pull-down 32-14
 - templates 32-15
- message
 - broadcast 8-20
 - dispatch 8-113
- Message Box 8-262
- Message Control Hook 10-18
- Message Filter Hook 10-20
- message processing
 - introduction 11-1
 - notation conventions 11-3

message processing (*continued*)

- types 11-1
- message queues 1-2
- message types 11-1
- messages
 - create queue 8-60
 - destroy queue 8-104
 - get one 8-183
 - peek 8-275
 - post 8-281
 - post queue 8-283
 - queues 1-2
 - send 8-437
 - wait for 8-567
- metaclass 9-1
- Metafile data format G-2
- metafile restrictions G-1
- metafiles
 - create new 5-57
 - delete 5-98
 - general rules G-1
 - load 5-165
 - play 5-193
 - query bits 5-321
 - query length 5-323
 - SAA-conforming 5-460, 5-465, 5-470, 5-472
 - save 5-389
- MIA_* values 17-2
- micro-presentation space 5-391, 5-474
- mini-icon size, how determined A-17
- MINIRECORDCORE A-69
- MIS_* values 17-2, 32-15
- MIT_* values 17-9, 17-12, 17-18
- mix
 - query 5-324
 - set 5-510
 - set background 5-415
 - set foreground 5-510
- MIXED strings 34-23
- MLECTLDATA A-69
- MLEMARGSTRUCT A-70
- MLEOVERFLOW A-71
- MLE_SEARCHDATA A-71
- MLM_CHARFROMLINE 18-8
- MLM_CLEAR 18-7
- MLM_COPY 18-7
- MLM_CUT 18-8
- MLM_DELETE 18-9
- MLM_DISABLEREFRESH 18-9
- MLM_ENABLEREFRESH 18-10
- MLM_EXPORT 18-11
- MLM_FORMAT 18-11
- MLM_IMPORT 18-12
- MLM_INSERT 18-13
- MLM_LINEFROMCHAR 18-13
- MLM_PASTE 18-14
- MLM_QUERYBACKCOLOR 18-14
- MLM_QUERYCHANGED 18-15
- MLM_QUERYFIRSTCHAR 18-16
- MLM_QUERYFONT 18-16
- MLM_QUERYFORMATLINELENGTH 18-17
- MLM_QUERYFORMATRECT 18-18
- MLM_QUERYFORMATTEXTLENGTH 18-17
- MLM_QUERYIMPORTEXPORT 18-18
- MLM_QUERYLINECOUNT 18-19
- MLM_QUERYLINELENGTH 18-19
- MLM_QUERYREADONLY 18-20

- MLM_QUERYSEL 18-20
- MLM_QUERYSELTEXT 18-21
- MLM_QUERYTABSTOP 18-22
- MLM_QUERYTEXTCOLOR 18-22
- MLM_QUERYTEXTLENGTH 18-23
- MLM_QUERYTEXTLIMIT 18-23
- MLM_QUERYUNDO 18-24
- MLM_QUERYWRAP 18-24
- MLM_RESETUNDO 18-25
- MLM_SEARCH 18-26
- MLM_SETBACKCOLOR 18-27
- MLM_SETCHANGED 18-28
- MLM_SETFIRSTCHAR 18-28
- MLM_SETFONT 18-29
- MLM_SETFORMATRECT 18-30
- MLM_SETIMPORTEXPORT 18-31
- MLM_SETREADONLY 18-32
- MLM_SETSEL 18-31
- MLM_SETTABSTOP 18-33
- MLM_SETTEXTCOLOR 18-32
- MLM_SETTEXTLIMIT 18-33
- MLM_SETWRAP 18-34
- MLM_UNDO 18-35
- MLS_* values 18-2
- MM_DELETEITEM 17-8
- MM_ENDMENUMODE 17-9
- MM_INSERTITEM 17-9
- MM_ISITEMVALID 17-10
- MM_ITEMIDFROMPOSITION 17-11
- MM_ITEMPOSITIONFROMID 17-11
- MM_QUERYITEM 17-12
- MM_QUERYITEMATTR 17-13
- MM_QUERYITEMCOUNT 17-13
- MM_QUERYITEMRECT 17-14
- MM_QUERYITEMTEXT 17-15
- MM_QUERYITEMTEXTLENGTH 17-15
- MM_QUERYSELITEMID 17-16
- MM_REMOVEITEM 17-17
- MM_SELECTITEM 17-18
- MM_SETITEM 17-19
- MM_SETITEMATTR 17-20
- MM_SETITEMHANDLE 17-20
- MM_SETITEMTEXT 17-21
- MM_STARTMENUMODE 17-22
- modal dialog
 - load and process 8-115
- Modify Path 5-170, 33-19
- monochrome devices 5-327
- Move 5-173
- Move to Next Character 8-268
- Move to Previous Character 8-285
- MPARAM A-72
- MPATH_* values 5-170
- MQINFO A-72
- MRESULT A-72
- MsgCtlHook 10-18
- MsgFilterHook 10-20
- MSGF_* values 10-20
- MS_* values 12-5, 17-1
- MTI A-72
- multi-line entry field control data 18-2
- multi-line entry field control window processing 18-1
- multiple-line statements 32-7
 - ACCELTABLE 32-9
 - ASSOCTABLE 32-10
 - DLGTEMPLATE 32-16
 - MENU 32-11

multiple-line statements (*continued*)

STRINGTABLE 32-7
WINDOWTEMPLATE 32-16
M_WPFileSystem * A-67
M_WPFolder * A-67
M_WPObject * A-67
M_WPPalette * A-67

N

No-Operation 33-19
nonstore attribute for segments
 modify (GpiSetSegmentAttrs) 5-539
notation conventions
 messages 11-3
notebook control window processing
 notification messages 25-3
 purpose 25-1
 styles 25-1
 window messages 25-4
NOTIFYDELTA A-73
NOTIFYDELTA data structure A-73
NOTIFYRECORDEMPHASIS A-73
NOTIFYRECORDEMPHASIS data structure A-73
NOTIFYRECORDENTER A-74
NOTIFYRECORDENTER data structure A-74
NOTIFYSCROLL A-74
NOTIFYSCROLL data structure A-74
NULL 1-1
NULLHANDLE 1-1

O

OBJCLASS A-75
OBJDATA A-75
Object classes 9-2
Offset Clip Region 5-175
Offset Element Pointer 5-177
Offset Rectangle 8-270
Offset Region 5-179
Open Clipboard 8-272
Open Device Context 2-9
open figure 5-20
Open Profile 6-3
Open Segment 5-181
Open Window Device Context 8-273
outline fonts 5-427, 5-430, 5-433, 5-438, 5-441, 5-445
Outline Path 5-184, 33-19
owner-notification messages 11-3
OWNERBACKGROUND A-75
OWNERBACKGROUND data structure A-75
OWNERITEM A-76
OWNERITEM data structure 12-75
 owneritem parameter 12-75, 24-6
 WM_DRAWITEM for container control 24-6
 WM_DRAWITEM for font dialog 12-75

P

PACCEL A-76
PACCELTABLE A-76
page viewport
 query 5-330
 set 5-516
PAGEINFO A-76
PAGESELECTNOTIFY A-78

paint

 begin 8-18
 end 8-141
Paint Region 5-186
palette
 animate 5-8
 create 5-81
 delete 5-100
 query 5-332
 query information 5-333
 realize 8-403
 select 5-396
 set entries 5-518
PALINFO A-78
PANOSE A-78, F-9
PAPSZ A-82
PARAM A-82
PARCPARAMS A-84
PAREABUNDLE A-84
parent/child/owner relationship 32-23
Partial Arc 5-188
Partial Arc at Current Position 33-20
Partial Arc at Given Position 33-20
path
 begin 5-19
 convert to region 5-191
 draw interior 5-142
 draw outline 5-184
 end 5-132
 fill 5-142
 modify 5-170
Path to Region 5-191
PATSYM_* values 5-335, 5-521
pattern
 query 5-335
pattern attribute (area) values 5-405, 5-463
patterns
 query reference point 5-336
 query set 5-337
 set 5-521
 set reference point 5-524
 set set 5-526
PBANDRECT A-84
PBITMAPINFO A-84
PBITMAPINFOHEADER A-84
PBITMAPINFOHEADER2 A-84
PBITMAPINFO2 A-84
PBOOKTEXT A-84
PBOOL A-84
PBUFFER A-84
PBUNDLE A-84
PBYTE A-84
PC VKEY 1-1
PCATCHBUF A-85
PCDATE A-85
PCELL A-85
PCH A-85
PCHAR A-85
PCHARBUNDLE A-85
PCLASSDETAILS A-85
PCLASSFIELDINFO A-85
PCLASSINFO A-85
PCNRDRAGINFO A-85
PCNRDRAGINIT A-85
PCNRDRAWITEMINFO A-85
PCNREDITDATA A-85
PCNRINFO A-85

- PCOLOR A-85
- PCONVCONTEXT A-85
- PCPTXT A-85
- PCREATEPARAMS A-85
- PCREATESTRUCT A-85
- PCTIME A-85
- PCURSORINFO A-85
- PDDEINIT A-85
- PDDESTRUCT A-86
- PDELENOTIFY A-86
- PDESKTOP A-86
- PDEVOPENDATA A-86
- PDEVOPENSTRUC A-86
- PDLGTEMPLATE A-86
- PDLGTITEM A-86
- PDRAGIMAGE A-86
- PDRAGINFO A-86
- PDRAGITEM A-86
- PDRAGTRANSFER A-86
- PDRIVDATA A-86
- PDRIVPROPS A-86
- Peek Message 8-275
- pel
 - query 5-338
 - set 5-528
- PENTRYFDATA A-86
- PERRINFO A-86
- PERRORID A-86
- PESCMODE A-86
- PFACENAMEDESC A-86
- PFATTRS A-86
- PFFDESCS A-87
- PFIELDINFO A-87
- PFIELDINFOINSERT A-87
- PFILEDLG A-87
- PFILEFINDBUF4 A-87
- PFIXED A-87
- PFN A-87
- PFNWP A-87
- PFOCAMETRICS type F-2
- PFONTDLG A-87
- PFONTMETRICS A-87
- PGRADIENTL A-87
- PHAB A-87
- PHBITMAP A-87
- PHCINFO A-87
- PHDC A-87
- PHelpINIT A-87
- PHelpSUBTABLE A-87
- PHelpTABLE A-87
- PHFIND A-87
- PHMF A-87
- PHMODULE A-87
- PHPAL A-87
- PHPROGARRAY A-88
- PHPROGRAM A-88
- PHPS A-88
- PHRGN A-88
- PHSEM A-88
- PHSWITCH A-88
- PHWND A-88
- PIBSTRUCT A-88
- pick aperture
 - query size 5-341
 - set size 5-531
- PICKAP_* values 5-531
- PICKSEL_* values 5-59, 5-63, 5-67

- PICONINFO A-89
- PICONPOS A-89
- PID A-89
- pie
 - segment 5-189
- PIMAGEBUNDLE A-89
- PIPT A-89
- PIX A-89
- PKERNINGPAIRS A-89
- Place Bitmap Reference 4-5
- Place Metafile Reference 4-21
- Play Metafile 5-193
- PLINEBUNDLE A-89
- PLONG A-89
- PL_ALTERED 12-3
- PMARGSTRUCT A-89
- PMARKERBUNDLE A-89
- PMATRIXLF A-89
- PMENUITEM A-89
- PMF_* values 5-193
- PMINIRECORDCORE A-89
- PMLE_SEARCHDATA A-89
- PMPARAM A-89
- PMQINFO A-89
- PMRESULT A-89
- PM_Q_* values A-26
- PM_* flags 8-275
- PM_* names H-1
- PM_* values 10-5, 10-13
- PNOTIFYDELTA A-90
- PNOTIFYRECORDEMPHASIS A-90
- PNOTIFYRECORDENTER A-90
- PNOTIFYSCROLL A-90
- POBJCLASS A-90
- POBJDATA A-90
- POBJECTS A-89
- Point Arc 5-199
- Point In Rectangle 8-289
- Point In Region 5-219
- Point Visible 5-221
- pointer
 - create 8-64
 - create indirect 8-66
 - destroy 8-107
 - draw 8-124
 - hide 8-520
 - implicit 1-1
 - load 8-248
 - query handle 8-342
 - query information 8-343
 - query position 8-345
 - set 8-484
 - set element 5-482
 - set position 8-486
 - show 8-520
- pointer file format D-2
- Pointer-Conversion Procedure 10-3
- POINTERINFO A-90
- pointing device
 - capture messages 8-442
- POINTL A-90
- points A-90
 - check whether visible 5-221
 - check whether within region 5-219
- Polyfillet 5-201
 - draw 5-201
 - sharp 5-204

- Polyfillet Sharp 5-204
- POLYGON A-91
- polygons 33-20
 - draw a set of 5-207
- Polyline 5-209
 - disjoint 5-211
 - draw 5-209
- Polyline Disjoint 5-211
- Polymarker 5-213
- Polyspline 5-215
- Pop 5-217, 33-21
- Pop-up Menu 8-277
- Post Device Modes 2-12
- Post Drag Message 3-24
- Post Message 8-281
- Post Queue Message 8-283
- POVERFLOW A-91
- POWNERBACKGROUND A-91
- POWNERITEM A-91
- PPAGEINFO A-91
- PPAGESELECTNOTIFY A-91
- PPALINFO A-89
- PPIBSTRUCT A-91
- PPID A-89
- PPOINTL A-91
- PPOINTS A-91
- PPOLYGON A-91
- PPRDINFO3 A-91
- PPRDRIVINFO A-91
- PPRESPARAMS A-91
- PPRINTDEST A-91
- PPRINTERINFO A-91
- PPRJINFO2 A-91
- PPRJINFO3 A-91
- PPROGCATEGORY A-91
- PPROGDETAILS A-91
- PPROGRAMENTRY A-92
- PPROGTITLE A-92
- PPROGTYPE A-92
- PPRPORTINFO A-92
- PPRPORTINFO1 A-92
- PPRQINFO3 A-92
- PPRQINFO6 A-92
- PPRQPROCINFO A-92
- PPSZ A-92
- PPVOID A-92
- PQMOPENDATA A-92
- PQMSG A-92
- PQUERYRECFROMRECT A-92
- PQUERYRECORDRECT A-92
- PRDINFO3 A-92
- PRDRIVINFO A-93
- PRECORDCORE A-93
- PRECORDINSERT A-93
- PRECTL A-94
- predefined control statements 32-24
- predefined window classes 32-23
- PRENDERFILE A-94
- Presentation Interface
 - initialize 8-199
- Presentation Manager
 - query environment 8-381
 - query revision level 8-381
 - query version 8-381
- presentation parameters 32-22
- presentation space
 - cache 8-18

- presentation space (*continued*)
 - cached 15-11
 - create 5-84
 - destroy 5-108
 - get a cache 8-190
 - micro 5-86, 8-119, 8-123, 8-128, 8-190
 - normal 8-119, 8-123, 8-128
 - options 5-84, 5-533
 - query 5-342
 - release cache 8-420
 - reset 5-382
 - restore 5-384
 - save 5-391
- presentation space options 5-84, 5-533
- PRESPARAMS A-94
- PrfCloseProfile 6-2
- PrfOpenProfile 6-3
- PRFPROFILE A-94
- PrfQueryProfile 6-5
- PrfQueryProfileData 6-7
- PrfQueryProfileInt 6-10
- PrfQueryProfileSize 6-12
- PrfQueryProfileString 6-14
- PrfReset 6-17
- PrfWriteProfileData 6-19
- PrfWriteProfileString 6-21
- PRGB2 A-94
- PRGNRECT A-94
- PRGN_* values 5-219
- primitives
 - set attributes for 5-404
- PRIM_* values 5-229, 5-271, 5-404, 5-462
- PRINTDEST A-94
- PRINTERINFO A-95
- PRJINFO2 A-96
- PRJINFO3 A-97
- procedures 10-1
 - dialog 10-2
 - window 10-4
- Process Modal Dialog 8-287
- profile
 - query string 6-14
- PROGCATEGORY A-99
- PROGDETAILS A-99
- PROGRAMENTRY A-100
- PROGTITLE A-100
- PROGTYPE A-100
- PROG_* values A-100
- prompted entry field control window processing 19-1
- PRPORTINFO A-101
- PRPORTINFO1 A-101
- PRQINFO3 A-101
- PRQINFO6 A-103
- PRQPROCINFO A-105
- PSBCDATA A-105
- PSEARCHSTRING A-105
- PSFACTORS A-105
- PSF_* values 8-169
- PSHORT A-105
- PSIZEF A-105
- PSIZEL A-105
- PSLDCDATA A-105
- PSTRL A-105
- PSTR16 A-105
- PSTR32 A-105
- PSTR64 A-105
- PSTR8 A-105

PSTYLECHANGE A-105
 PSWBLOCK A-106
 PSWCNTRL A-106
 PSWENTRY A-106
 PSWP A-106
 PSZ A-106
 PS * values 5-84, 5-342, 5-533
 PTID A-106
 PTRACKINFO A-106
 PTREEITEMDESC A-106
 PUCCHAR A-106
 pull-down menus 32-14
 PULONG A-106
 PUSEITEM A-106
 PUSERBUTTON A-106
 Push and Set Arc Parameters 33-23
 Push and Set Background Color 33-23
 Push and Set Background Indexed Color 33-24
 Push and Set Background Mix 33-25
 Push and Set Character Angle 33-26
 Push and Set Character Break Extra 33-26
 Push and Set Character Cell 33-27
 Push and Set Character Direction 33-28
 Push and Set Character Extra 33-28
 Push and Set Character Precision 33-29
 Push and Set Character Set 33-30
 Push and Set Character Shear 33-30
 Push and Set Color 33-31
 Push and Set Current Position 33-32
 Push and Set Extended Color 33-32
 Push and Set Fractional Line Width 33-33
 Push and Set Indexed Color 33-34
 Push and Set Individual Attribute 33-35
 Push and Set Line End 33-36
 Push and Set Line Join 33-36
 Push and Set Line Type 33-37
 Push and Set Line Width 33-38
 Push and Set Marker Cell 33-39
 Push and Set Marker Precision 33-40
 Push and Set Marker Set 33-40
 Push and Set Marker Symbol 33-41
 Push and Set Mix 33-41
 Push and Set Model Transform 33-42
 Push and Set Pattern Reference Point 33-43
 Push and Set Pattern Set 33-44
 Push and Set Pattern Symbol 33-44
 Push and Set Pick Identifier 33-45
 Push and Set Stroke Line Width 33-46
 Push and Set Text Alignment 33-47
 Push and Set Viewing Window 33-48
 PUSHORT A-106
 Put Data 5-223
 PU * values 5-84, 5-533
 PVIEWFONTCELLSIZE A-106
 PVIEWSIZECOUNT A-106
 PVIS * values 5-221
 PVOID A-106
 PVSCDATA A-106
 PVSDRAGINFO A-106
 PVSDRAGINIT A-106
 PVSTEXT A-106
 PWNDPARAMS A-106
 PWPOINT A-106

Q

QCD_LCT_* values 5-264
 QFC_* values 15-16
 QF_* values 5-299
 QLCT_* values 5-313
 QMOPENSTRUC A-107
 QMSG 11-1, A-108
 QS_* values 8-352
 Query Accelerator Table 8-291
 Query Active Window 8-293
 Query Anchor Block 8-294
 Query Arc Parameters 5-226
 Query Atom Length 8-295
 Query Atom Name 8-297
 Query Atom Usage 8-299
 Query Attribute Mode 5-228
 Query Attributes 5-229
 Query Background Color 5-231
 Query Background Mix 5-232
 Query Bit-Map Bits 5-233
 Query Bit-Map Dimension 5-236
 Query Bit-Map Handle 5-239
 Query Bit-Map Info Header 5-237
 Query Bit-Map Parameters 5-240
 Query Boundary Data 5-242
 Query Capture 8-302
 Query Character Angle 5-244
 Query Character Box 5-246
 Query Character Break Extra 5-248
 Query Character Direction 5-249
 Query Character Extra 5-250
 Query Character Mode 5-251
 Query Character Set 5-252
 Query Character Shear 5-253
 Query Character String Positions 5-255
 Query Character String Positions At 5-257
 Query Checkstate of Button 8-300
 Query Class Information 8-303
 Query Class Name 8-305
 Query Class Pointer-Conversion Procedure 8-307
 Query Clip Box 5-259
 Query Clip Region 5-261
 Query Clipboard Data 8-308
 Query Clipboard Format Information 8-310
 Query Clipboard Owner 8-312
 Query Clipboard Viewer 8-313
 Query Code Page 5-268, 8-314
 Query Code Page List 8-315
 Query Color 5-262
 Query Color Data 5-264
 Query Color Index 5-266
 Query Current Position 5-269
 Query Cursor Information 8-316
 Query Default Arc Parameters 5-270
 Query Default Attributes 5-271
 Query Default Graphics Character Box 5-275
 Query Default Tag 5-277
 Query Default View Matrix 5-273
 Query Default Viewing Limits 5-278
 Query Desktop Background 8-317
 Query Desktop Window 8-319
 Query Device 5-279
 Query Device Bit-Map Formats 5-280
 Query Device Capabilities 2-15
 Query Device Names 2-21
 Query Dialog Item Short 8-321

- Query Dialog Item Text 8-323
- Query Dialog Item Text Length 8-325
- Query Draw Control 5-282
- Query Drawing Mode 5-284
- Query Edit Mode 5-285
- Query Element 5-286
- Query Element Pointer 5-288
- Query Element Type 5-290
- Query Face String 5-292
- Query Focus 8-327
- Query Font Action 5-294
- Query Font File Descriptions 5-295
- Query Font Metrics 5-297
- Query Font Width Table 5-372
- Query Fonts 5-299
- Query Full Font File Descriptions 5-301
- Query Graphics Field 5-303
- Query Hardcopy Caps 2-24
- Query Help Instance 8-328
- Query Initial Segment Attributes 5-304
- Query Kerning Pairs 5-306
- Query Line End 5-308
- Query Line Join 5-309
- Query Line Type 5-310
- Query Line Width 5-311
- Query Line Width Geom 5-312
- Query Listbox Item Text 8-331
- Query Listbox Item Text Length 8-333
- Query Logical Color Table 5-313
- Query Logical Font 5-315
- Query Marker 5-317
- Query Marker Box 5-318
- Query Marker Set 5-320
- Query Message Position 8-336
- Query Message Time 8-338
- Query Metafile Bits 5-321
- Query Metafile Length 5-323
- Query Mix 5-324
- Query Model Transform Matrix 5-325
- Query Nearest Color 5-327
- Query Number Set Identifiers 5-329
- Query Object Window 8-340
- Query Page Viewport 5-330
- Query Palette 5-332
- Query Palette Info 5-333
- Query Pattern 5-335
- Query Pattern Reference Point 5-336
- Query Pattern Set 5-337
- Query Pel 5-338
- Query Pick Aperture Position 5-340
- Query Pick Aperture Size 5-341
- Query Pointer 8-342
- Query Pointer Information 8-343
- Query Pointer Position 8-345
- Query Presentation Parameter 8-347
- Query Presentation Space 5-342
- Query Profile 6-5
- Query Profile Data 6-7
- Query Profile Integer 6-10
- Query Profile Size 6-12
- Query Profile String 6-14
- Query Queue Information 8-350
- Query Queue Status 8-352
- Query Real Colors 5-343
- Query Region Box 5-345
- Query Region Rectangles 5-347
- Query RGB Color 5-349

- Query Segment Attributes 5-351
- Query Segment Names 5-353
- Query Segment Priority 5-355
- Query Segment Transform Matrix 5-357
- Query Session Title 8-355
- Query Set Identifiers 5-359
- Query Stop Draw 5-362
- Query Switch Entry 8-357
- Query Switch Handle 8-358
- Query Switch List 8-360
- Query System Atom Table 8-372
- Query System Color 8-362
- Query System Modal Window 8-364
- Query System Pointer 8-365
- Query System Value 8-368
- Query Tag 5-363
- Query Task Title 8-375
- Query Task Window Size and Position 8-373
- Query Text Alignment 5-364
- Query Text Box 5-365
- Query the Selected Item in Listbox 8-335
- Query Update Rectangle 8-377
- Query Update Region 8-379
- Query Version 8-381
- Query Viewing Limits 5-368
- Query Viewing Transform Matrix 5-370
- Query Window 8-382
- Query Window Device Context 8-384
- Query Window Enabled State 8-228
- Query Window Handle From Device Context 8-572
- Query Window Handle From Identifier 8-574
- Query Window Long 8-398
- Query Window Model 8-385
- Query Window Pointer 8-390
- Query Window Pointer-Conversion Procedure 8-397
- Query Window Position 8-386
- Query Window Process 8-388
- Query Window Rectangle 8-392
- Query Window Short 8-400
- Query Window Showing 8-230
- Query Window Text 8-394
- Query Window Text Length 8-396
- Query Window Visibility 8-232
- Query Workplace Object Handle 8-402
- QUERYRECFROMRECT A-108
- QUERYRECFROMRECT data structure A-108
- QUERYRECORDRECT A-109
- QUERYRECORDRECT data structure A-109
- queue
 - query information 8-350
 - query status 8-352
- QV_* values 8-381
- QWL_USER in containers 24-1
- QWL_* values 8-398
- QWS_* values 8-400
- QW_*values 8-382

R

- radio button 13-1
- raster fonts 5-427, 5-430, 5-433, 5-438, 5-441, 5-445
- Realize Palette 8-403
- RECORDCORE A-110
- RECORDINSERT A-111
- RECORDINSERT data structure A-111
- RECORDITEM A-111
- rectangle

- rectangle (*continued*)
 - calculate frame 8-22
 - check whether visible 5-376
 - check whether within region 5-374
 - compare for equality 8-148
 - convert to graphic 8-258
 - copy 8-39
 - draw border 8-121
 - draw interior 8-121
 - exclude from clipping region 5-140
 - fill 8-154
 - inflate 8-197
 - intersect 8-205
 - intersect clip 5-155
 - invalidate 8-207
 - invert 8-211
 - query if point within 8-289
 - query update 8-377
 - set coordinates 8-489
 - set empty 8-491
 - subtract 8-538
 - validate 8-560
- Rectangle In Region 5-374
- Rectangle Visible 5-376
- RECTDIR_* values A-114
- RECTL A-112
- region
 - query box 5-345
 - query rectangles 5-347
- regions
 - check if identical 5-134
 - check whether point within 5-219
 - check whether rectangle within 5-374
 - combine 5-49
 - create 5-88
 - destroy 5-110
 - frame 5-146
 - invalidate 8-209
 - move 5-179
 - offset 5-179
 - paint 5-186
 - set 5-536
 - validate 8-562
- Register User Data Type 8-408
- Register User Message 8-415
- Register User Message Hook 10-21
- Register Window Class 8-405
- Register Workplace Object Class 8-407
- RegisterUserMsg 10-21
- Relative Line at Current Position 33-22
- Relative Line at Given Position 33-22
- Release Hook 8-418
- Release Presentation Space 3-44, 8-420
- Remove Dynamics 5-378
- Remove Presentation Parameter 8-422
- Remove Switch Entry 8-424
- RENDERFILE A-112
- Replace Workplace Object Class 8-426
- Request Mutex Semaphore 8-427
- reserved messages 12-1
- Reset Boundary Data 5-381
- reset options 5-194
- Reset Presentation Manager 6-17
- Reset Presentation Space 5-382
- resource
 - load string from 8-251
- resource definitions 32-2

- resource file specification 32-27
- resource files
 - definitions 32-2
 - introduction 32-1
 - source file specification 32-27
 - syntax definitions 32-1
- resource script file
 - specification 32-2
- resource script file specification
 - keyboard resources 32-18
 - user-defined resources 32-3
- resource statements
 - ACCELTABLE 32-9
 - ASSOCTABLE 32-10
 - dialog template 32-16
 - directives 32-4
 - DLGTEMPLATE 32-16
 - MENU item definition 32-13
 - MENU statement 32-11
 - multiple-line 32-7
 - single line 32-2
 - STRINGTABLE 32-7
 - user-defined 32-3
 - window template 32-16
 - WINDOWTEMPLATE 32-16
- Restore Presentation Space 5-384
- Restore Window Position 8-429
- RES_* values 5-194
- RGB 5-77, A-113
- RGB (red-green-blue) 5-264, 5-343, 5-453, 8-362
 - query color 5-349
- RGB2 A-113
- RGNRECT A-114
- RGN_* values 5-140, 5-155, 5-345, 5-451, 8-379
- Right cursor key 8-547
- Roman text 5-435
- ROP_* values 5-24, 5-112, 5-567
- Rotate Transform 5-386
- RRGN_* values 5-374
- RT_* values 32-27
- RUM_* values 8-415
- RVIS_* values 5-376

S

- SAA-conforming metafiles 5-475
- Save Metafile 5-389
- Save Presentation Space 5-391
- Save Window Position 8-430
- SBCDATA A-114
- SBCS 34-23
- SBMP_* values 8-194
- SBM_QUERYPOS 20-4
- SBM_QUERYRANGE 20-4
- SBM_SETPOS 20-5
- SBM_SETSCROLLBAR 20-6
- SBM_SETTHUMBSize 20-7
- SBS_* values 20-1
- SB_* values 12-38, 12-68, 28-2, 28-5
- Scale Matrix 5-393
- SCP_* values 5-448
- scroll bar control data 20-1
- scroll bar control window.processing 20-1
- scroll bar styles 20-1
- Scroll Window 8-432
- SC_* values 15-21
- SDW_* values 5-362, 5-546

SEARCHSTRING A-115
 SEARCHSTRING data structure A-115
 SEGEM_* values 5-285, 5-480
 segment attributes
 chained 5-539
 detectability 5-539
 highlight 5-539
 nonstore 5-539
 store 5-539
 transformability 5-539
 visibility 5-539
 Segment Characteristics 33-22
 segments
 add comment 5-51
 call matrix 5-31
 close current 5-47
 correlate 5-67
 correlate chain 5-59
 correlate section of chain 5-63
 delete all 5-104
 delete retained 5-102
 draw 5-123
 draw chain 5-117
 draw section of chain 5-121
 get graphic data from 5-150
 open 5-181
 query attributes 5-351
 query initial attributes 5-304
 query names 5-353
 query priority 5-355
 query transform matrix 5-357
 return last error during drawing 5-138
 set attributes 5-538
 set initial attributes 5-488
 set priority 5-541
 set transform matrix 5-543
 Select Palette 5-396
 Send Drag Message 3-45
 Send Message 8-437
 Send Message Hook 10-23
 Send Message to Dialog Item 8-435
 SendMsgHook 10-23
 SEPARATOR menu item 32-15
 session title
 query 8-355
 Set Accelerator Table 8-439
 Set Active Window 8-441
 Set Arc Parameters 5-398, 33-23
 Set Attribute Mode 5-401
 Set Attributes 5-404
 Set Background Color 5-412, 33-23
 Set Background Indexed Color 33-24
 Set Background Mix 5-415, 33-25
 Set Bit Map 5-418
 Set Bit-Map Bits 5-420
 Set Bit-Map Dimension 5-423
 Set Bit-Map Identifier 5-425
 Set Capture 8-442
 Set Character Angle 5-427, 33-26
 Set Character Box 5-430
 Set Character Break Extra 5-433, 33-26
 Set Character Cell 33-27
 Set Character Direction 5-435, 33-28
 Set Character Extra 5-438, 33-28
 Set Character Mode 5-440
 Set Character Precision 33-29
 Set Character Set 5-443, 33-30
 Set Character Shear 5-445, 33-30
 Set Checkstate of Button 8-30
 Set Class Message Interest 8-444
 Set Class Pointer-Conversion Procedure 8-447
 Set Clip Path 5-448, 33-31
 Set Clip Region 5-451
 Set Clipboard Data 8-449
 Set Clipboard Owner 8-452
 Set Clipboard Viewer 8-454
 Set Code Page 5-456, 8-456
 Set Color 5-453, 33-31
 Set Color of Text 4-26
 Set Current Position 5-458, 33-32
 Set Default Arc Parameters 5-460
 Set Default Attributes 5-462
 Set Default Tag 5-470
 Set Default View Matrix 5-467
 Set Default Viewing Limits 5-472
 Set Desktop Background 8-457
 Set Dialog Item Short 8-459
 Set Dialog Item Text 8-461
 Set Drag Image 3-48
 Set Draw Control 5-474
 Set Drawing Mode 5-477
 Set Edit Mode 5-480
 Set Element Pointer 5-482
 Set Element Pointer At Label 5-484
 Set Extended Color 33-32
 Set File Icon 8-463
 Set Focus 8-464
 Set Fractional Line Width 33-33
 Set Graphics Field 5-486
 Set Hook 8-466
 set identifier
 delete 5-106
 Set Indexed Color 33-34
 Set Individual Attribute 33-35
 Set Initial Segment Attributes 5-488
 Set Keyboard State Table 8-468
 Set Line End 5-491, 33-36
 Set Line Join 5-493, 33-36
 Set Line Type 5-495, 33-37
 Set Line Width 5-498, 33-38
 Set Line Width Geom 5-500
 Set Listbox Item Text 8-470
 Set Marker 5-502
 Set Marker Box 5-504
 Set Marker Cell 33-39
 Set Marker Precision 33-40
 Set Marker Set 5-506, 33-40
 Set Marker Symbol 33-41
 Set Menu Item Text 8-472
 Set Message Interest 8-473
 Set Message Mode 8-476
 Set Metafile Bits 5-508
 Set Mix 5-510, 33-41
 Set Model Transform 33-42
 Set Model Transform Matrix 5-513
 Set Multiple Window Positions 8-478
 Set Object Data 8-480
 Set Owner 8-481
 Set Page Viewport 5-516
 Set Palette Entries 5-518
 Set Parent 8-482
 Set Pattern 5-521
 Set Pattern Reference Point 5-524, 33-43
 Set Pattern Set 5-526, 33-44

- Set Pattern Symbol 33-44
- Set Pel 5-528
- Set Pick Identifier 33-45
- Set Pick-Aperture Position 5-530
- Set Pick-Aperture Size 5-531
- Set Pointer 8-484
- Set Pointer Position 8-486
- Set Pointing Device Pointer 3-53
- Set Presentation Parameter 8-487
- Set Presentation Space 5-533
- Set Rectangle 8-489
- Set Rectangle Empty 8-491
- Set Region 5-536
- Set Segment Attributes 5-538
- Set Segment Boundary 33-45
- Set Segment Priority 5-541
- Set Segment Transform Matrix 5-543
- Set Stop Draw 5-546
- Set Stroke Line Width 33-46
- Set Synchronization Mode 8-492
- Set System Colors 8-494
- Set System Modal Window 8-500
- Set System Value 8-502
- Set Tag 5-548
- Set Text Alignment 5-550, 33-47
- Set Values in DRAGITEM 3-50
- Set Viewing Limits 5-553
- Set Viewing Transform 33-48
- Set Viewing Transform Matrix 5-555
- Set Viewing Window 33-48
- Set Window Enabled State 8-135
- Set Window Pointer-Conversion Procedure 8-514
- Set Window Position 8-506
- Set Window Text 8-512
- Set Window Word Bits 8-504
- Set Window Word Long 8-515
- Set Window Word Short 8-517
- Set Window Words Pointer 8-510
- SFACTORS A-115
- SHANDLE A-116
- Sharp Fillet at Current Position 33-50
- Sharp Fillet at Given Position 33-50
- SHE_* values A-101
- SHORT A-116
- Show Cursor 8-518
- Show Pointer 8-520
- Show Tracking Rectangle 8-522
- Show Window 8-523
- Shutdown System 8-525
- single-byte character set 1-6
- single-byte character sets 34-23
- SIZEF A-116
- SIZEL A-116
- SLDCDATA A-116
- SLDCDATA data structure A-116
- slider control window processing
 - data structures 26-3
 - notification messages 26-4
 - purpose 26-1
 - styles 26-1
 - window messages 26-7
- SLM_ADDDETENT 26-7
- SLM_QUERYDETENTPOS 26-7
- SLM_QUEYS CALETEXT 26-8
- SLM_QUEYS LIDERINFO 26-9
- SLM_QUERYTICKPOS 26-11
- SLM_QUERYTICKSIZE 26-11
- SLM_REMOVEDETENT 26-12
- SLM_SETSCALETEXT 26-13
- SLM_SETSLIDERINFO 26-13
- SLM_SETTICKSIZE 26-15
- SLS_* values 26-1
- SMHSTRUCT A-117
- SMIM_* values 8-444, 8-473
- SMI_* values 8-444, 8-473
- SM_QUERYHANDLE 22-3
- SM_SETHANDLE 22-4
- Sound Alarm 8-11
- source resource file 32-27
- SPBM_OVERRIDESETLIMITS 21-3
- SPBM_QUERYLIMITS 21-4
- SPBM_QUERYVALUE 21-4
- SPBM_SETARRAY 21-6
- SPBM_SETCURRENTVALUE 21-6
- SPBM_SETLIMITS 21-7
- SPBM_SETMASTER 21-8
- SPBM_SETTEXTLIMIT 21-9
- SPBM_SPINDOWN 21-9
- SPBM_SPINUP 21-10
- Specify Text Font 4-29
- Specify Text Font Style 4-32
- spin button control window processing 21-1
 - notification message 21-2
 - purpose 21-1
 - styles 21-1
- SplControlDevice 7-2
- SplCopyJob 7-5
- SplCreateDevice 7-7
- SplCreateQueue 7-10
- SplDeleteDevice 7-14
- SplDeleteJob 7-16
- SplDeleteQueue 7-18
- SplEnumDevice 7-20
- SplEnumDriver 7-23
- SplEnumJob 7-26
- SplEnumPort 7-29
- SplEnumPrinter 7-32
- SplEnumQueue 7-35
- SplEnumQueueProcessor 7-39
- SPLERR A-117
- SplHoldJob 7-42
- SplHoldQueue 7-44
- SplPurgeQueue 7-46
- SplQmAbort 7-48
- SplQmAbortDoc 7-49
- SplQmClose 7-50
- SplQmEndDoc 7-51
- SplQmOpen 7-53
- SplQmStartDoc 7-55
- SplQmWrite 7-57
- SplQueryDevice 7-59
- SplQueryJob 7-62
- SplQueryQueue 7-66
- SplReleaseJob 7-70
- SplReleaseQueue 7-72
- SplSetDevice 7-74
- SplSetJob 7-77
- SplSetQueue 7-81
- SPL_* values 7-51, 7-53
- Spool File Close 7-50
- spooler
 - control device 7-2
 - copy job 7-5
 - create device 7-7

- spooler (*continued*)
 - create queue 7-10
 - delete device 7-14
 - delete job 7-16
 - delete queue 7-18
 - enumerate device 7-20
 - enumerate driver 7-23, 7-29
 - enumerate job 7-26
 - enumerate printer 7-32
 - enumerate queue 7-35
 - enumerate queue processor 7-39
 - hold job 7-42
 - hold queue 7-44
 - purge queue 7-46
 - query device 7-59
 - query job 7-62
 - query queue 7-66
 - queue manager abort 7-48
 - queue manager abort document 7-49
 - queue manager close 7-50
 - queue manager end document 7-51
 - queue manager open 7-53
 - queue manager start document 7-55
 - queue manager write 7-57
 - release job 7-70
 - release queue 7-72
 - set device 7-74
 - set job information 7-77
 - set queue 7-81
- Spooler Control Device 7-2
- Spooler Copy Job 7-5
- Spooler Create Device 7-7
- Spooler Create Queue 7-10
- Spooler Delete Device 7-14
- Spooler Delete Job 7-16
- Spooler Delete Queue 7-18
- Spooler Enumerate Device 7-20
- Spooler Enumerate Driver 7-23
- Spooler Enumerate Job 7-26
- Spooler Enumerate Port 7-29
- Spooler Enumerate Print Destinations 7-32
- Spooler Enumerate Queue 7-35
- Spooler Enumerate Queue Processor 7-39
- Spooler File Abort 7-48
- Spooler File Abort Document 7-49
- Spooler File End Document 7-51
- Spooler File Open 7-53
- Spooler File Start Document 7-55
- Spooler File Write 7-57
- Spooler Hold Job 7-42
- Spooler Hold Queue 7-44
- Spooler Purge Queue 7-46
- Spooler Query Device 7-59
- Spooler Query Job 7-62
- Spooler Query Queue 7-66
- Spooler Release Job 7-70
- Spooler Release Queue 7-72
- Spooler Set Device 7-74
- Spooler Set Job 7-77
- Spooler Set Queue 7-81
- SPTR_* values 8-365
- SS_* values 22-1
- standard bit-map formats D-1
- Standard File Dialog 8-152
- Standard File Dialog Default Procedure 8-87
- Standard Font Dialog 8-163
- Standard Font Dialog Default Procedure 8-88
- Start Timer 8-529
- static control data 22-2
- static control styles 22-1
- static control window processing 22-1
- Stop Timer 8-531
- storage mapping of data types 1-6
- store attribute for segments
 - modify (GpiSetSegmentAttrs) 5-539
- Store Window Position 8-533
- string
 - convert to uppercase 8-556
- string handle
 - create 3-5
 - delete 3-10, 3-11
- strings
 - load from resource 8-251
 - substitute 8-536
- STRINGTABLE statement 32-7
- Stroke Path 5-558
- STRUCT A-117
- structures A-1
- STR16 A-117
- STR32 A-117
- STR64 A-117
- STR8 A-117
- STYLECHANGE A-117
- Subclass Window 8-534
- submenus 32-14
- Substitute Strings 8-536
- Subtract Rectangle 8-538
- suppression options 5-194
- SUP_* values 5-194
- SV_* values
 - effect on container icon size A-17
 - effect on container mini-icon size A-17
- SWBLOCK A-118
- SWCCTRL A-118
- SWENTRY A-119
- Switch To Program 8-540
- SWL_* values A-119
- SWP A-119
- SWP_* values 8-386, 8-506, 12-69, A-120
- SW_* options 8-432
- SYSCLR_* indexes 8-494
- SYSINF_* values 8-381
- system color
 - query 8-362
 - set 8-494
- system pointer
 - query 8-365
- system value
 - query 8-368
 - set 8-502

T

- tag
 - query 5-363
 - query default 5-277
 - set 5-548
- TA_* values 5-550, 5-551
- TBM_QUERYHILITE 23-3
- TBM_SETHILITE 23-3
- templates
 - dialog 32-19
 - format 32-15
 - menus 32-15

- Terminate 8-542
- Terminate Application 8-544
- text
 - draw 8-126
 - query alignment 5-364
 - query box 5-365
 - set alignment 5-550
- TF_* values A-121
- ThunkProc 10-3
- TID A-120
- timer
 - start 8-529
- title bar
 - control data 23-1
 - control window processing 23-1
 - style 23-1
- TRACKINFO A-120
- tracking rectangle
 - hide 8-522
 - show 8-522
- transform matrix
 - query model 5-325
 - rotate 5-386
 - scale 5-393
 - set model 5-513
 - translate 5-560
- transformability attribute for segments
 - modify (GpiSetSegmentAttrs) 5-539
- transforms
 - set viewing 5-555
- TRANSFORM_* values 5-31, 5-386, 5-393, 5-467, 5-513, 5-543, 5-555, 5-560
- Translate Accelerator 8-550
- Translate Character with Code Page 8-40
- Translate Matrix 5-560
- Translate String with Code Page 8-42
- TREEITEMDESC A-122
- triplets G-2
- TXTBOX_* values 5-366

U

- UCHAR A-122
- ULONG A-122
- Union Rectangle 8-552
- Unload Fonts 5-563
- Unload Public Fonts 5-565
- Up cursor key 8-547
- update region
 - exclude 8-150
 - query 8-379
- Update Window 8-554
- Uppercase Character 8-558
- Uppercase String 8-556
- USEITEM A-122
- user-defined resources 32-3
- USERBUTTON A-122
- USHORT A-123

V

- Validate Rectangle 8-560
- Validate Region 8-562
- value set control window processing
 - data structures 27-4
 - notification messages 27-5
 - purpose 27-1

- value set control window processing (*continued*)
 - styles 27-1
 - window messages 27-8
- Verify Given Rendering Mechanism and Format 3-57
- Verify Native Rendering Mechanism and Format 3-55
- Verify True Type of Dragged Object 3-59
- Verify Type of Dragged Object 3-61
- Verify Types 3-63
- VGA 2-19
- VIA_* values
 - querying item attributes 27-9
 - setting item attributes 27-15
- view matrix
 - query default 5-273
- viewing limits
 - query 5-368
 - query default 5-278
 - set 5-553
- viewing transform
 - set default 5-467
- viewing transforms
 - query 5-370
- VIEWITEM A-123
- viewports
 - query page 5-330
- VIOFONTCELLSIZE A-123
- VIOSIZECOUNT A-123
- virtual key definitions I-1
- visibility attribute for segments
 - modify (GpiSetSegmentAttrs) 5-539
- VK_* values 8-176, A-1
- VM_QUERYITEM 27-8
- VM_QUERYITEMATTR 27-9
- VM_QUERYMETRICS 27-11
- VM_QUERYSELECTEDITEM 27-12
- VM_SELECTITEM 27-12
- VM_SETITEM 27-13
- VM_SETITEMATTR 27-14
- VM_SETMETRICS 27-16
- VOID A-123
- VSCDATA A-123
- VSCDATA data structure A-123
- VSDRAGINFO A-123
- VSDRAGINFO data structure A-123
- VSDRAGINIT A-124
- VSTEXT A-124
- VS_* values 27-1

W

- Wait Event Semaphore 8-565
- Wait Message 8-567
- Wait MuxWait Semaphore or Message 8-569
- WA_* values 8-11
- WCS_* values 8-35
- WC_* classes 8-398
- WC_* values 11-2, 23-1
- WinAddAtom 8-7
- WinAddSwitchEntry 8-9
- WinAlarm 8-11
- WinAssociateHelpInstance 8-13
- WinBeginEnumWindows 8-16
- WinBeginPaint 8-18
- WinBroadcastMsg 8-20
- WinCalcFrameRect 8-22
- WinCallMsgFilter 8-24
- WinCancelShutdown 8-26

- WinChangeSwitchEntry 8-28
- WinCheckButton 8-30
- WinCheckMenuItem 8-32
- WinCloseClipbrd 8-34
- WinCompareStrings 8-35
- WinCopyAccelTable 8-37
- WinCopyRect 8-39
- WinCpTranslateChar 8-40
- WinCpTranslateString 8-42
- WinCreateAccelTable 8-44
- WinCreateAtomTable 8-46
- WinCreateCursor 8-48
- WinCreateDlg 8-50
- WinCreateFrameControls 8-52
- WinCreateHelpInstance 8-54
- WinCreateHelpTable 8-56
- WinCreateMenu 8-58
- WinCreateMsgQueue 8-60
- WinCreateObject 8-62
- WinCreatePointer 8-64
- WinCreatePointerIndirect 8-66
- WinCreateStdWindow 8-68
- WinCreateSwitchEntry 8-72
- WinCreateWindow 8-74
- WinDdeInitiate 8-78
- WinDdePostMsg 8-80
- WinDdeRespond 8-83
- WinDefDlgProc 8-85
- WinDefFileDlgProc 8-87
- WinDefFontDlgProc 8-88
- WinDefWindowProc 8-89
- WinDeleteAtom 8-91
- WinDeleteLboxItem 8-93
- WinDeleteLibrary 8-95
- WinDeleteProcedure 8-96
- WinDeregisterObjectClass 8-97
- WinDestroyAccelTable 8-98
- WinDestroyAtomTable 8-99
- WinDestroyCursor 8-101
- WinDestroyHelpInstance 8-102
- WinDestroyMsgQueue 8-104
- WinDestroyObject 8-106
- WinDestroyPointer 8-107
- WinDestroyWindow 8-109
- WinDismissDlg 8-111
- WinDispatchMsg 8-113
- WinDlgBox 8-115
- window
 - create 8-74
 - destroy 8-109
 - query 8-382
 - query active 8-293
 - query class name 8-305
 - query desktop 8-319
 - query device context for 8-384
 - query handle from device context 8-572
 - query pointer 8-390
 - query position 8-386
 - query size 8-386
 - query text 8-394
 - query text length 8-396
 - query unsigned long integer value of 8-398
 - query unsigned short integer value of 8-400
 - register class of 8-405
 - scroll 8-432
 - set message interest 8-473
 - set multiple positions 8-478
- window (*continued*)
 - set owner 8-481
 - set position 8-506
 - set to system modal 8-500
 - update 8-554
- window class
 - set message interest 8-444
- window class styles 12-1
- Window From Point 8-576
- window list
 - remove entry 8-424
- Window List title
 - query 8-375
- Window Procedure 10-4
- window processing
 - button control 13-1
 - combo box control 19-1
 - container control 24-1
 - control 11-2
 - default 11-1, 12-1
 - entry field control 14-1
 - frame control 15-1
 - language support 12-80
 - list box control 16-1
 - menu control 17-1
 - multi-line entry field control 18-1
 - notebook control 25-1
 - prompted entry field control 19-1
 - scroll bar control 20-1
 - slider control 26-1
 - spin button control 21-1
 - static control 22-1
 - value set control 27-1
- Window Start Application 8-526
- windows
 - create standard 8-68
 - create standard frame controls 8-52
 - define procedure 10-4
 - enable update 8-137
 - find descendant 8-576
 - get maximum position 8-179
 - get minimum position 8-181
 - get multiples from identities 8-266
 - invoke default procedure 8-89
 - is handle valid 8-226
 - map points 8-260
 - open device context 8-273
 - process message box 8-262
 - query class information 8-303
 - query descendancy 8-213
 - query enabled state 8-228
 - query handle from identifier 8-574
 - query is child 8-213
 - query object 8-340
 - query rectangle 8-392
 - query system modal 8-364
 - query visibility 8-232
 - set active 8-441
 - set enabled state 8-135
 - set parent 8-482
 - set text 8-512
 - set visibility state 8-137, 8-523
 - show 8-523
 - start flashing 8-158
 - stop flashing 8-158
- WINDOWTEMPLATE statement 32-16
- WinDrawBitmap 8-118

WinDrawBorder	8-121	WinLoadProcedure	8-250
WinDrawPointer	8-124	WinLoadString	8-251
WinDrawText	8-126	WinLockVisRegions	8-253
WinEmptyClipbrd	8-130	WinLockWindowUpdate	8-255
WinEnableControl	8-131	WinMakePoints	8-257
WinEnableMenuItem	8-132	WinMakeRect	8-258
WinEnablePhysInput	8-134	WinMapDlgPoints	8-259
WinEnableWindow	8-135	WinMapWindowPoints	8-260
WinEnableWindowUpdate	8-137	WinMessageBox	8-262
WinEndEnumWindows	8-139	WinMultWindowFromIDs	8-266
WinEndPaint	8-141	WinNextChar	8-268
WinEnumClipbrdFmts	8-143	WinOffsetRect	8-270
WinEnumDlgItem	8-145	WinOpenClipbrd	8-272
WinEnumObjectClasses	8-147	WinOpenWindowDC	8-273
WinEqualRect	8-148	WinPeekMsg	8-275
WinExcludeUpdateRegion	8-150	WinPopupMenu	8-277
WinFileDialog	8-152	WinPostMsg	8-281
WinFillRect	8-154	WinPostQueueMsg	8-283
WinFindAtom	8-156	WinPrevChar	8-285
WinFlashWindow	8-158	WinProcessDlg	8-287
WinFocusChange	8-160	WinPtInRect	8-289
WinFontDlg	8-163	WinQueryAccelTable	8-291
WinFreeErrorInfo	8-165	WinQueryActiveWindow	8-293
WinFreeFileDialogList	8-166	WinQueryAnchorBlock	8-294
WinFreeFileIcon	8-168	WinQueryAtomLength	8-295
WinGetClipPS	8-169	WinQueryAtomName	8-297
WinGetCurrentTime	8-171	WinQueryAtomUsage	8-299
WinGetDlgMsg	8-172	WinQueryButtonCheckstate	8-300
WinGetErrorInfo	8-175	WinQueryCapture	8-302
WinGetKeyState	8-176	WinQueryClassInfo	8-303
WinGetLastError	8-178	WinQueryClassName	8-305
WinGetMaxPosition	8-179	WinQueryClassThunkProc	8-307
WinGetMinPosition	8-181	WinQueryClipbrdData	8-308
WinGetMsg	8-183	WinQueryClipbrdFmtInfo	8-310
WinGetNextWindow	8-186	WinQueryClipbrdOwner	8-312
WinGetPhysKeyState	8-188	WinQueryClipbrdViewer	8-313
WinGetPS	8-190	WinQueryCp	8-314
WinGetScreenPS	8-192	WinQueryCpList	8-315
WinGetSysBitmap	8-194	WinQueryCursorInfo	8-316
WinInflateRect	8-197	WinQueryDesktopBkgnd	8-317
WinInitialize	8-199	WinQueryDesktopWindow	8-319
WinInSendMessage	8-201	WinQueryDlgItemShort	8-321
WinInsertLBoxItem	8-203	WinQueryDlgItemText	8-323
WinIntersectRect	8-205	WinQueryDlgItemTextLength	8-325
WinInvalidRect	8-207	WinQueryFocus	8-327
WinInvalidRegion	8-209	WinQueryHelpInstance	8-328
WinInvertRect	8-211	WinQueryLBoxCount	8-330
WinIsChild	8-213	WinQueryLBoxItemText	8-331
WinIsControlEnabled	8-214	WinQueryLBoxItemTextLength	8-333
WinIsMenuItemChecked	8-216	WinQueryLBoxSelectedItem	8-335
WinIsMenuItemEnabled	8-218	WinQueryMsgPos	8-336
WinIsMenuItemValid	8-220	WinQueryMsgTime	8-338
WinIsPhysInputEnabled	8-222	WinQueryObject	8-402
WinIsRectEmpty	8-223	WinQueryObjectWindow	8-340
WinIsThreadActive	8-224	WinQueryPointer	8-342
WinIsWindow	8-226	WinQueryPointerInfo	8-343
WinIsWindowEnabled	8-228	WinQueryPointerPos	8-345
WinIsWindowShowing	8-230	WinQueryPresParam	8-347
WinIsWindowVisible	8-232	WinQueryQueueInfo	8-350
WinLoadAccelTable	8-234	WinQueryQueueStatus	8-352
WinLoadDlg	8-236	WinQuerySessionTitle	8-355
WinLoadFileIcon	8-239	WinQuerySwitchEntry	8-357
WinLoadHelpTable	8-241	WinQuerySwitchHandle	8-358
WinLoadLibrary	8-243	WinQuerySwitchList	8-360
WinLoadMenu	8-244	WinQuerySysColor	8-362
WinLoadMessage	8-246	WinQuerySysModalWindow	8-364
WinLoadPointer	8-248	WinQuerySysPointer	8-365

WinQuerySystemAtomTable	8-372	WinSetSysValue	8-502
WinQuerySysValue	8-368	WinSetWindowBits	8-504
WinQueryTaskSizePos	8-373	WinSetWindowPos	8-506
WinQueryTaskTitle	8-375	WinSetWindowPtr	8-510
WinQueryUpdateRect	8-377	WinSetWindowText	8-512
WinQueryUpdateRegion	8-379	WinSetWindowThunkProc	8-514
WinQueryVersion	8-381	WinSetWindowULong	8-515
WinQueryWindow	8-382	WinSetWindowUShort	8-517
WinQueryWindowDC	8-384	WinShowCursor	8-518
WinQueryWindowModel	8-385	WinShowPointer	8-520
WinQueryWindowPos	8-386	WinShowTrackRect	8-522
WinQueryWindowProcess	8-388	WinShowWindow	8-523
WinQueryWindowPtr	8-390	WinShutdownSystem	8-525
WinQueryWindowRect	8-392	WinStartApp	8-526
WinQueryWindowText	8-394	WinStartTimer	8-529
WinQueryWindowTextLength	8-396	WinStopTimer	8-531
WinQueryWindowThunkProc	8-397	WinStoreWindowPos	8-533
WinQueryWindowULong	8-398	WinSubclassWindow	8-534
WinQueryWindowUShort	8-400	WinSubstituteStrings	8-536
WinRealizePalette	8-403	WinSubtractRect	8-538
WinRegisterClass	8-405	WinSwitchToProgram	8-540
WinRegisterObjectClass	8-407	WinTerminate	8-542
WinRegisterUserDatatype	8-408	WinTerminateApp	8-544
WinRegisterUserMsg	8-415	WinTrackRect	8-546
WinReleaseHook	8-418	WinTranslateAccel	8-550
WinReleasePS	8-420	WinUnionRect	8-552
WinRemovePresParam	8-422	WinUpdateWindow	8-554
WinRemoveSwitchEntry	8-424	WinUpper	8-556
WinReplaceObjectClass	8-426	WinUpperChar	8-558
WinRequestMutexSem	8-427	WinValidateRect	8-560
WinRestoreWindowPos	8-429	WinValidateRegion	8-562
WinSaveWindowPos	8-430	WinWaitEventSem	8-565
WinScrollWindow	8-432	WinWaitMsg	8-567
WinSendDlgItemMsg	8-435	WinWaitMuxWaitSem	8-569
WinSendMsg	8-437	WinWindowFromDC	8-572
WinSetAccelTable	8-439	WinWindowFromID	8-574
WinSetActiveWindow	8-441	WinWindowFromPoint	8-576
WinSetCapture	8-442	WM_ACTIVATE	8-109, 8-508, 12-3
WinSetClassMsgInterest	8-444	WM_ACTIVATE (in Frame Controls)	15-6
WinSetClassThunkProc	8-447	WM_ACTIVATE (Language Support Dialog)	12-83
WinSetClipbrdData	8-449	WM_ACTIVATE (Language Support Window)	12-80
WinSetClipbrdOwner	8-452	WM_ADJUSTFRAMEPOS	15-6
WinSetClipbrdViewer	8-454	WM_ADJUSTWINDOWPOS	8-508, 12-5
WinSetCp	8-456	WM_APPTERMINATENOTIFY	12-4
WinSetDesktopBkgnd	8-457	WM_BEGINDRAG	12-6
WinSetDlgItemShort	8-459	WM_BEGINSELECT	12-7
WinSetDlgItemText	8-461	WM_BUTTON1CLICK	12-7
WinSetFileIcon	8-463	WM_BUTTON1DBLCLK	12-10
WinSetFocus	8-464	WM_BUTTON1DBLCLK (in Frame Controls)	15-7
WinSetHook	8-466	WM_BUTTON1DBLCLK (in Multiline Entry Fields)	18-36
WinSetKeyboardStateTable	8-468	WM_BUTTON1DOWN	12-13
WinSetLboxItemText	8-470	WM_BUTTON1DOWN (in Frame Controls)	15-8
WinSetMenuitemText	8-472	WM_BUTTON1DOWN (in Multiline Entry Fields)	18-36
WinSetMsgInterest	8-473	WM_BUTTON1MOTIONEND	12-14
WinSetMsgMode	8-476	WM_BUTTON1MOTIONSTART	12-14
WinSetMultWindowPos	8-478	WM_BUTTON1UP	12-19
WinSetObjectData	8-480	WM_BUTTON1UP (in Frame Controls)	15-8
WinSetOwner	8-481	WM_BUTTON1UP (in Multiline Entry Fields)	18-37
WinSetParent	8-482	WM_BUTTON2CLICK	12-8
WinSetPointer	8-484	WM_BUTTON2DBLCLK	12-11
WinSetPointerPos	8-486	WM_BUTTON2DBLCLK (in Frame Controls)	15-7
WinSetPresParam	8-487	WM_BUTTON2DOWN	12-15
WinSetRect	8-489	WM_BUTTON2DOWN (in Frame Controls)	15-8
WinSetRectEmpty	8-491	WM_BUTTON2MOTIONEND	12-16
WinSetSynchroMode	8-492	WM_BUTTON2MOTIONSTART	12-16
WinSetSysColors	8-494	WM_BUTTON2UP	12-20
WinSetSysModalWindow	8-500	WM_BUTTON2UP (in Frame Controls)	15-9

WM_BUTTON3CLICK 12-9
 WM_BUTTON3DBLCLK 12-12
 WM_BUTTON3DOWN 12-17
 WM_BUTTON3MOTIONEND 12-18
 WM_BUTTON3MOTIONSTR 12-18
 WM_BUTTON3UP 12-21
 WM_CALCFRAMERECT 12-22
 WM_CALCFRAMERECT (in Frame Controls) 15-9
 WM_CALCVALIDRECTS 12-22
 WM_CHAR 12-24
 WM_CHAR (Default Dialogs) 12-70
 WM_CHAR (in Entry Fields) 14-12
 WM_CHAR (in Frame Controls) 15-9
 WM_CHAR (in List Boxes) 16-15
 WM_CHAR (in Multiline Entry Fields) 18-37
 WM_CHAR (in Notebook Controls) 25-18
 WM_CHAR (in Slider Controls) 26-16
 WM_CHAR (in Value Set Controls) 27-17
 WM_CHORD 12-25
 WM_CLOSE 12-26
 WM_CLOSE (Default Dialogs) 12-71
 WM_CLOSE (in Frame Controls) 15-10
 WM_COMMAND 11-3, 12-27, 15-10
 WM_COMMAND (Default Dialogs) 12-71
 WM_COMMAND (in Button Controls) 13-3
 WM_COMMAND (in Menu Controls) 17-4
 WM_CONTEXTMENU 12-28
 WM_CONTROL 11-3, 12-28
 WM_CONTROL (in Button Controls) 13-3
 WM_CONTROL (in Combination Boxes) 19-3
 WM_CONTROL (in Container Controls) 24-4
 WM_CONTROL (in Entry Fields) 14-3
 WM_CONTROL (in List Boxes) 16-2
 WM_CONTROL (in Multiline Entry Fields) 18-3
 WM_CONTROL (in Notebook Controls) 25-3
 WM_CONTROL (in Slider Controls) 26-4
 WM_CONTROL (in Spin Button Controls) 21-2
 WM_CONTROL (in Value Set Controls) 27-5
 WM_CONTROL (Language Support Dialog) 12-83
 WM_CONTROL (Language Support Window) 12-80
 WM_CONTROLPOINTER 12-29
 WM_CONTROLPOINTER (in Container Controls) 24-5
 WM_CONTROLPOINTER (in Notebook Controls) 25-19
 WM_CONTROLPOINTER (in Slider Controls) 26-4
 WM_CONTROLPOINTER (in Value Set Controls) 27-6
 WM_CREATE 12-29
 WM_DDE_ACK 30-1
 WM_DDE_ADVISE 30-2
 WM_DDE_DATA 30-3
 WM_DDE_EXECUTE 30-3
 WM_DDE_INITIATE 30-5
 WM_DDE_INITIATEACK 30-5
 WM_DDE_POKE 30-6
 WM_DDE_REQUEST 30-7
 WM_DDE_TERMINATE 30-8
 WM_DDE_UNADVISE 30-9
 WM_DESTROY 8-109, 12-30
 WM_DESTROYCLIPBOARD 28-1
 WM_DRAWCLIPBOARD 28-2
 WM_DRAWITEM 12-31
 WM_DRAWITEM (in Container Controls) 24-6
 WM_DRAWITEM (in Font Dialog) 12-75
 WM_DRAWITEM (in Frame Controls) 15-10
 WM_DRAWITEM (in List Boxes) 16-3
 WM_DRAWITEM (in Menu Controls) 17-4
 WM_DRAWITEM (in Notebook Controls) 25-20

WM_DRAWITEM (in Slider Controls) 26-5
 WM_DRAWITEM (in Value Set Controls) 27-6
 WM_ENABLE 12-31
 WM_ENABLE (in Button Controls) 13-10
 WM_ENABLE (in Multiline Entry Fields) 18-40
 WM_ENDDRAG 12-32
 WM_ENDSELECT 12-33
 WM_ERASEBACKGROUND 15-10
 WM_ERASEWINDOW 12-33
 WM_ERROR 12-34
 WM_FLASHWINDOW 15-11
 WM_FOCUSCHANGE 12-34
 WM_FOCUSCHANGE (in Frame Controls) 15-12
 WM_FORMATFRAME 12-35
 WM_FORMATFRAME (in Frame Controls) 15-12
 WM_HELP 11-3, 12-36
 WM_HELP (in Button Controls) 13-4
 WM_HELP (in Menu Controls) 17-5
 WM_HITTEST 12-37
 WM_HSCROLL 12-38
 WM_HSCROLL (in Horizontal Scroll Bars) 20-3
 WM_HSCROLLCLIPBOARD 28-2
 WM_INITDLG 12-38
 WM_INITDLG (Default Dialogs) 12-71
 WM_INITMENU 12-39
 WM_INITMENU (in Frame Controls) 15-13
 WM_INITMENU (in Menu Controls) 17-5
 WM_JOURNALNOTIFY 12-39
 WM_MATCHMNEMONIC 12-40
 WM_MATCHMNEMONIC (Default Dialogs) 12-71
 WM_MATCHMNEMONIC (in Button Controls) 13-10
 WM_MATCHMNEMONIC (in Static Controls) 22-4
 WM_MEASUREITEM 12-41
 WM_MEASUREITEM (in Frame Controls) 15-13
 WM_MEASUREITEM (in List Boxes) 16-4
 WM_MEASUREITEM (in Menu Controls) 17-5
 WM_MENUEND 12-41
 WM_MENUEND (in Menu Controls) 17-6
 WM_MENUSELECT 12-42
 WM_MENUSELECT (in Frame Controls) 15-13
 WM_MENUSELECT (in Menu Controls) 17-6
 WM_MINMAXFRAME 12-42
 WM_MINMAXFRAME (in Frame Controls) 15-4
 WM_MOUSEMOVE 12-43
 WM_MOUSEMOVE (in Multiline Entry Fields) 18-40
 WM_MOVE 8-508, 12-44
 WM_NEXTMENU 12-44
 WM_NEXTMENU (in Frame Controls) 15-14
 WM_NEXTMENU (in Menu Controls) 17-7
 WM_NULL 12-45
 WM_OPEN 12-45
 WM_OWNERPOSCCHANGE 15-14
 WM_PACTIVATE 12-46
 WM_PAINT 12-47
 WM_PAINT (in Frame Controls) 15-15
 WM_PAINT (Language Support Window) 12-80
 WM_PAINT (Language Support Dialog) 12-83
 WM_PAINTCLIPBOARD 28-3
 WM_PCONTROL 12-47
 WM_PPAINT 12-48
 WM_PPAINT (Language Support Dialog) 12-84
 WM_PPAINT (Language Support Window) 12-81
 WM_PRESPARAMCHANGED 12-48
 WM_PRESPARAMCHANGED (in Container Controls) 24-52
 WM_PRESPARAMCHANGED (in Notebook Controls) 25-21

WM_PRESPARAMCHANGED (in Slider Controls) 26-17
 slider control 26-17
 value set control 27-18
 WM_PRESPARAMCHANGED (in Value Set Controls) 27-18
 WM_PSETFOCUS 12-49
 WM_PSIZE 12-49
 WM_PSYSCOLORCHANGE 12-50
 WM_QUERYACCELTABLE 12-50
 WM_QUERYBORDERSIZE 15-15
 WM_QUERYCONVERTPOS 12-51
 WM_QUERYCONVERTPOS (in Button Controls) 13-10
 WM_QUERYCONVERTPOS (in Entry Fields) 14-13
 WM_QUERYCONVERTPOS (in Frame Controls) 15-16
 WM_QUERYCONVERTPOS (in List Boxes) 16-15
 WM_QUERYCONVERTPOS (in Menu Controls) 17-23
 WM_QUERYCONVERTPOS (in Scroll Bars) 20-8
 WM_QUERYCONVERTPOS (in Static Controls) 22-5
 WM_QUERYCONVERTPOS (in Title Bar Controls) 23-4
 WM_QUERYDLGCODE 12-72
 WM_QUERYFOCUSCHAIN 15-16
 WM_QUERYFRAMECTLCOUNT 15-17
 WM_QUERYFRAMEINFO 15-18
 WM_QUERYHELPINFO 12-52
 WM_QUERYICON 15-18
 WM_QUERYTRACKINFO 12-52
 WM_QUERYWINDOWPARAMS 12-53
 WM_QUERYWINDOWPARAMS (in Button Controls) 13-11
 WM_QUERYWINDOWPARAMS (in Entry Fields) 14-13
 WM_QUERYWINDOWPARAMS (in Frame Controls) 15-19
 WM_QUERYWINDOWPARAMS (in List Boxes) 16-16
 WM_QUERYWINDOWPARAMS (in Menu Controls) 17-23
 WM_QUERYWINDOWPARAMS (in Multiline Entry Fields) 18-41
 WM_QUERYWINDOWPARAMS (in Scroll Bars) 20-8
 WM_QUERYWINDOWPARAMS (in Slider Controls) 26-18
 slider control 26-18
 value set control 27-19
 WM_QUERYWINDOWPARAMS (in Static Controls) 22-5
 WM_QUERYWINDOWPARAMS (in Title Bars) 23-4
 WM_QUERYWINDOWPARAMS (in Value Set Controls) 27-19
 WM_QUIT 12-53
 WM_REALIZEPALETTE 12-54
 WM_RENDERALLFMTS 8-109, 28-4
 WM_RENDERFMT 28-4
 WM_SAVEAPPLICATION 12-55
 WM_SEM1 12-55
 WM_SEM2 12-56
 WM_SEM3 12-56
 WM_SEM4 12-57
 WM_SETACCELTABLE 12-57
 WM_SETBORDERSIZE 15-19
 WM_SETFOCUS 12-58
 WM_SETFOCUS (Language Support Dialog) 12-84
 WM_SETFOCUS (Language Support Window) 12-81
 WM_SETHELPINFO 12-58
 WM_SETICON 15-20
 WM_SETSELECTION 12-59
 WM_SETWINDOWPARAMS 12-60
 WM_SETWINDOWPARAMS (in Button Controls) 13-11
 WM_SETWINDOWPARAMS (in Entry Fields) 14-13
 WM_SETWINDOWPARAMS (in Frame Controls) 15-20
 WM_SETWINDOWPARAMS (in List Boxes) 16-16
 WM_SETWINDOWPARAMS (in Menu Controls) 17-23
 WM_SETWINDOWPARAMS (in Multiline Entry Fields) 18-42
 WM_SETWINDOWPARAMS (in Scroll Bars) 20-8
 WM_SETWINDOWPARAMS (in Slider Controls) 26-19
 slider control 26-19
 value set control 27-20
 WM_SETWINDOWPARAMS (in Static Controls) 22-5
 WM_SETWINDOWPARAMS (in Title Bar Controls) 23-4
 WM_SETWINDOWPARAMS (in Value Set Controls) 27-20
 WM_SHOW 12-60
 WM_SINGLESELECT 12-61
 WM_SIZE 8-508, 12-61
 WM_SIZE (in Frame Controls) 15-20
 WM_SIZE (in Notebook Controls) 25-22
 WM_SIZE (in Value Set Controls) 27-20
 WM_SIZE (Language Support Dialog) 12-84
 WM_SIZE (Language Support Window) 12-81
 WM_SIZECLIPBOARD 28-5
 WM_SUBSTITUTESTRING 12-62
 WM_SYSCOLORCHANGE 12-63
 WM_SYSCOLORCHANGE (Language Support Dialog) 12-85
 WM_SYSCOLORCHANGE (Language Support Window) 12-82
 WM_SYSCOMMAND 12-63, 13-4, 15-21, 17-7
 WM_SYSCOMMAND (in Title Bar Controls) 23-2
 WM_SYSVALUECHANGED 12-64
 WM_TEXTEDIT 12-65
 WM_TIMER 12-65
 WM_TRACKFRAME 12-66
 WM_TRACKFRAME (in Frame Controls) 15-22
 WM_TRACKFRAME (in Title Bar Controls) 23-2
 WM_TRANSLATEACCEL 12-67
 WM_TRANSLATEACCEL (in Frame Controls) 15-23
 WM_TRANSLATEMNEMONIC 12-67
 WM_TRANSLATEMNEMONIC (in Frame Controls) 15-23
 WM_UPDATEFRAME 12-68
 WM_UPDATEFRAME (in Frame Controls) 15-23
 WM_VSCROLL 12-68
 WM_VSCROLL (in Vertical Scroll Bars) 20-3
 WM_VSCROLLCLIPBOARD 28-5
 WM_WINDOWPOSCHANGED 12-69
 WM * messages 8-352
 WNDPARAMS A-125
 WndProc 10-4
 World Coordinates Bit Bit 5-567
 wpAddClockAlarmPage 9-53
 wpAddClockDateTimePage 9-54
 wpAddClockView1Page 9-55
 wpAddClockView2Page 9-56
 wpAddCountryDatePage 9-57
 wpAddCountryNumbersPage 9-58
 wpAddCountryPage 9-59
 wpAddCountryTimePage 9-60
 wpAddDesktopLockup1Page 9-61
 wpAddDesktopLockup2Page 9-62
 wpAddDesktopLockup3Page 9-63
 wpAddDiskDetailsPage 9-64
 wpAddFileMenuPage 9-65
 wpAddFileTypePage 9-66
 wpAddFile1Page 9-67
 wpAddFile2Page 9-68
 wpAddFile3Page 9-69
 wpAddFolderBackgroundPage 9-70
 wpAddFolderIncludePage 9-71
 wpAddFolderSortPage 9-72
 wpAddFolderView1Page 9-73

- wpAddFolderView2Page 9-74
- wpAddFolderView3Page 9-75
- wpAddKeyboardMappingsPage 9-76
- wpAddKeyboardSpecialNeedsPage 9-77
- wpAddKeyboardTimingPage 9-78
- wpAddMouseMappingsPage 9-79
- wpAddMouseTimingPage 9-80
- wpAddMouseTypePage 9-81
- wpAddObjectGeneralPage 9-82
- wpAddProgramAssociationPage 9-83, 9-84
- wpAddProgramPage 9-85, 9-86
- wpAddProgramSessionPage 9-87, 9-88
- wpAddSettingsPages 9-89
- wpAddSoundWarningBeepPage 9-90
- wpAddSystemConfirmationPage 9-91
- wpAddSystemLogoPage 9-92
- wpAddSystemPrintScreenPage 9-93
- wpAddSystemWindowPage 9-94
- wpAddToObjUseList 9-95
- wpAllocMem 9-97
- WPClock * A-125
- wpClose 9-98
- wpclsCreateDefaultTemplates 9-240
- wpclsFindObjectEnd 9-241
- wpclsFindObjectFirst 9-242
- wpclsFindObjectNext 9-244
- wpclsInitData 9-246
- wpclsMakeAwake 9-247
- wpclsNew 9-249
- wpclsQueryDefaultHelp 9-251
- wpclsQueryDefaultView 9-252
- wpclsQueryDetails 9-253
- wpclsQueryDetailsInfo 9-254
- wpclsQueryEditString 9-257
- wpclsQueryError 9-258
- wpclsQueryFolder 9-259
- wpclsQueryIcon 9-260
- wpclsQueryIconData 9-261
- wpclsQueryInstanceFilter 9-262
- wpclsQueryInstanceType 9-263
- wpclsQueryObject 9-264
- wpclsQueryOpenFolders 9-265
- wpclsQuerySettingsPageSize 9-266
- wpclsQueryStyle 9-267
- wpclsQueryTitle 9-268
- wpclsSetError 9-269
- wpclsUnInitData 9-270
- wpCnrInsertObject 9-99
- wpCnrRemoveObject 9-101
- wpCnrSetEmphasis 9-102
- wpConfirmDelete 9-103
- wpCopiedFromTemplate 9-104
- wpCopyObject 9-105
- WPCountry * A-125
- wpCreateFromTemplate 9-106
- wpCreateShadowObject 9-107
- WPDataFile * A-125
- wpDelete 9-108
- wpDeleteAllJobs 9-109
- wpDeleteContents 9-110
- wpDeleteFromObjUseList 9-111
- wpDeleteJob 9-112
- WPDesktop * A-125
- WPDisk * A-125
- wpDisplayHelp 9-113
- wpDoesObjectMatch 9-114
- wpDragCell 9-115

- wpDraggedOverObject 9-116
- wpDragOver 9-118
- wpDrop 9-119
- wpDroppedOnObject 9-120
- wpEditCell 9-121
- wpEndConversation 9-122
- WPFileSystem * A-125
- wpFilterPopupMenu 9-123
- wpFindUseltem 9-125
- WPFolder * A-125
- wpFormatDragItem 9-126
- wpFree 9-127
- wpFreeMem 9-128
- wpHide 9-129
- wpHideFldrRunObjs 9-130
- wpHoldJob 9-131
- wpHoldPrinter 9-132
- wpInitData 9-133
- wpInsertPopupMenuItems 9-134
- wpInsertSettingsPage 9-136
- wpIsCurrentDesktop 9-137
- WPJob * A-126
- WPKeyboard * A-126
- wpMenuItemHelpSelected 9-138
- wpMenuItemSelected 9-139
- wpModifyPopupMenu 9-140
- WPMouse * A-126
- wpMoveObject 9-141
- WPM_* values A-125
- WPObject * A-126
- WPOINT A-126
- wpOpen 9-142
- wpPaintCell 9-143
- WPPalette * A-126
- wpPopulate 9-144
- WPPrinter * A-126
- wpPrintJobNext 9-145
- wpPrintMetaFile 9-146
- wpPrintObject 9-147
- wpPrintPifFile 9-148
- wpPrintPlainTextFile 9-149
- wpPrintPrinterSpecificFile 9-150
- wpPrintUnknownFile 9-151
- WPProgramFile * A-126
- WPProgramGroup * A-126
- WPProgram * A-126
- wpQueryAssociationFilter 9-152, 9-153
- wpQueryAssociationType 9-154, 9-155
- wpQueryComputerName 9-156
- wpQueryConfirmations 9-157
- wpQueryContent 9-158
- wpQueryDefaultHelp 9-159
- wpQueryDefaultView 9-160
- wpQueryDetailsData 9-161
- wpQueryError 9-163
- wpQueryFldrAttr 9-164
- wpQueryFldrDetailsClass 9-165
- wpQueryFldrFlags 9-166
- wpQueryFldrFont 9-167
- wpQueryHandle 9-168
- wpQueryIcon 9-169
- wpQueryIconData 9-170
- wpQueryLogicalDrive 9-171
- wpQueryNextIconPos 9-172
- wpQueryPaletteHelp 9-173
- wpQueryPaletteInfo 9-174
- wpQueryPrinterName 9-175

- wpQueryProgDetails 9-176, 9-177
- wpQueryRealName 9-178
- wpQueryRootFolder 9-179
- wpQueryShadowedObject 9-180
- wpQueryStyle 9-181
- wpQueryTitle 9-182
- wpQueryType 9-183
- wpRedrawCell 9-184
- wpRefresh 9-185
- wpRegisterView 9-186
- wpReleaseJob 9-187
- wpReleasePrinter 9-188
- wpRender 9-189
- wpRenderComplete 9-190
- wpRestore 9-191
- wpRestoreData 9-192
- wpRestoreLong 9-193
- wpRestoreState 9-194
- wpRestoreString 9-195
- WPRootFolder * A-126
- wpSaveData 9-196
- wpSaveDeferred 9-197
- wpSaveImmediate 9-198
- wpSaveLong 9-199
- wpSaveState 9-200
- wpSaveString 9-201
- wpScanSetupString 9-202
- wpSetAssociationFilter 9-204, 9-205
- wpSetAssociationType 9-206, 9-207
- wpSetComputerName 9-208
- wpSetDefaultHelp 9-209
- wpSetDefaultPrinter 9-210
- wpSetDefaultView 9-211
- wpSetError 9-212
- wpSetFldrAttr 9-213
- wpSetFldrDetailsClass 9-214
- wpSetFldrFlags 9-215
- wpSetFldrFont 9-216
- wpSetIcon 9-217
- wpSetIconData 9-218
- wpSetNextIconPos 9-219
- wpSetPaletteInfo 9-220
- wpSetPrinterName 9-221
- wpSetProgDetails 9-222, 9-223
- wpSetRealName 9-224
- wpSetShadowTitle 9-225
- wpSetStyle 9-226
- wpSetTitle 9-227
- wpSetType 9-228
- wpSetup 9-229
- wpSetupCell 9-233
- WPSHADOW * A-126
- wpShowPalettePointer 9-234
- WPSOUND * A-126
- WPSPOOLER * A-126
- WPSRCLASSBLOCK* A-126
- wpStartJobAgain 9-235
- wpSwitchTo 9-236
- WPSYSTEM * A-127
- wpUnInitData 9-238
- wpUnlockObject 9-237
- WRECT A-127
- Write Profile Data 6-19
- Write Profile String 6-21
- WS_* values 8-190, 12-2

X

- XYF_* values A-128
- XYWINSIZE A-127

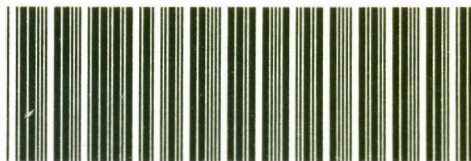


® IBM, OS/2 and Operating System/2 are
registered trademarks of
International Business Machines Corporation



© IBM Corp. 1992
International Business
Machines Corporation

Printed in the
United States of America
All Rights Reserved
10G6264



S10G-6264-00



P10G6264